VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Server pro podporu výuky teorie her

Supporting Server for Games Theory Teaching

2014 Bc. Marek Záškodný

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Marek Záškodný**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Server pro podporu výuky teorie her
Supporting Server for Games Theory Teaching

Zásady pro vypracování:

V rámci série diplomových prací vzniká server pro podporu výuky teorie her. Student si na něm bude moci zahrát vybrané hry proti počítači uplatňujícímu optimální strategii, ale také si bude moci zobrazit různé výpočty a řešení úloh z oblasti teorie her. Cílem této diplomové práce je přidání základních úloh týkajících se her dvou hráčů s nenulovým součtem.

Konkrétní zásady:
1. Seznamte se stručně s typy her, které se v teorii her zkoumají.
2. Podrobně se zaměřte na hry dvou hráčů s nenulovým součtem.
3. S kolegy pracujícími také na vytvoření tohoto serveru si dohodněte softwarovou architekturu a vzhled, aby server působil jednotně.
4. Vytvořte část zabývající se hrami s nenulovým součtem.
5. Pro matice výplatních funkcí her s nenulovým součtem bude program umět hledat různé typy strategií obou hráčů při možné spolupráci i bez spolupráce, převádět mezi grafovou a maticovou specifikací hry atd.
6. Umožněte, aby si uživatel mohl několik zvolených her zahrát proti počítači.

Seznam doporučené odborné literatury:

Thomas S. Ferguson: Game Theory, elektronický výukový text dostupný na adrese http://www.math.ucla.edu/~tom/math167.html (především Part 3)

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Martin Kot, Ph.D.**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2014

.....

Rád bych na tomto místě poděkoval svému vedoucímu diplomové práce panu Ing. Martinu Kotovi, Ph.D., jehož připomínky byly pro mě podnětem a přispěly ke zkvalitnění této práce.

Abstrakt

Diplomová práce se zabývá vytvořením serveru pro podporu výuky teorie her. Teorie her je disciplína aplikované matematiky, která se zabývá rozhodováním v konfliktních situacích a uplatňuje se v mnoha oblastech lidské činnosti. Cílem této práce je rozšířit stávající sérii diplomových prací o hry dvou hráčů s nenulovým součtem. Na serveru mají studenti možnost nastudovat si veškerou teorii k pochopení dané problematiky a dále zde bude možnost zahrát si několik typů her proti počítači, který bude hrát náhodnou nebo optimální strategii. U všech typů her je možnost zobrazit si maticovou a grafickou specifikaci hry. Dále se pak podle typu dané hry zobrazí optimální strategie, cena hry, strategie hrozby, Nashovy rovnovážné body, Paretovo optimum, atd.

Klíčová slova: teorie her, nenulový součet, C#, Visual Studio, webová aplikace

Abstract

This thesis deals with the creation of a supporting server for Game Theory teaching. The Game Theory is discipline of applied mathematics, that deals with decisions in conflict situations and applies in many areas of human activity. The aim of this work is to extend the existing series of theses by non-zero sum games for two players. On the server, students have the opportunity to study the entire theory to understand the issue and there is the opportunity to play several types of games against a computer that will play random or optimal strategy. For all types of games there will be the opportunity to display a matrix and extensive specification of games. Furthermore, under the type of the game displays the optimal strategy, the price of the game, threat strategies, Nash equilibria, Pareto optimum, etc.

Keywords: game theory, non-zero-sum, C#, Visual Studio, web application

Seznam použitých zkratek a symbolů

IT	–	Informační technologie
ASP	–	Active Server Pages
DLL	–	Dynamic-link library
VB	–	Visual Basic
UI	–	User interface
GUI	–	Graphical user interface
CSS	–	Cascading Style Sheets
AJAX	–	Asynchronous JavaScript and XML

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Člověk se denně ocitá v situacích, kdy se musí rozhodnout pro vhodné jednání nebo postup, aby dosáhl pokud možno co nejlepšího výsledku. Když to závisí jen na něm samém, případně na nějakých dalších okolnostech, které může předvídat, ale které jsou nezávislé na zásahu kohokoli jiného, pak je rozhodovací situace poměrně jednoduchá. Složitější je dosažení úspěchu v případech, kdy to závisí i na jednání nějaké další osoby nebo více osob. Vzniká tak přirozeně množství dalších vlivů, okolností a variant jejich jednání, se kterými je nutné počítat pro dosažení cíle. Jejich identifikace a analýza je nedílnou součástí teorie her.

V tomto smyslu se pak nejedná o běžný pojem „hry“, ale o řešení situací v oblastech obchodu, financí, pojišťovnictví, průmyslu, dopravy, zemědělství, skladování, vědy, vojenství i politiky. Situace vznikající v těchto i mnoha dalších oblastech pak budeme při jejich analýze a řešení nazývat „hrami“ a jejich účastníky „hráči“.

Obsahem analýzy jednotlivých her musí být určení jejich pravidel, definice cílů, identifikace hráčů, popis možných vlivů a případných konfliktů, informací nutných pro rozhodování a jejich specifikací, ovlivňujících výsledek.

Na základě uvedené analýzy je pak úkolem vytvořit pro praktické využití určitý návod nebo strategii, které by hráčům umožnily dosažení optimálního řešení problému v dané hře.

Cílem této práce je vytvoření sytému, v kterém je možné analyzovat řešení různých typů her, zahrát si hry proti počítači a vyzkoušet si simulaci většího počtu her. Ukázka serveru pro podporu výuky, který jsem vytvořil, se nachází na adrese: *http://teh-vsb.aspone.cz/*.

Celá práce je rozdělena na několik částí. Začíná teoretickým úvodem do Teorie her. Další část je zaměřena na samotný server pro podporu výuky od analýzy požadavků, návrhu serveru, až po samotnou implemetaci. Poslední část je věnována testování a nasazení serveru do reálného provozu a zhodnocení výsledků práce.

2 Teorie her

Teorie her je oblast aplikované matematiky zabývající se rozhodováním v konfliktních situacích. Hra je matematický model rozhodovací situace, jehož výsledek závisí na rozhodnutí nejméně dvou různých subjektů. S takovými situacemi se můžeme setkat v mnoha oblastech našeho života. V podstatě kdekoliv, kde dochází ke střetu zájmů. Matematické modely se pak snaží tyto konfliktní situace analyzovat a pomocí výpočtů nalézt co nejoptimálnější strategie pro konkrétní subjekty daných konfliktů. Teorie her je velmi obsáhlá a rozsah oblastí, do kterých zasahuje je neobyčejně široký. Můžeme se s ní setkat v ekonomii, vojenství, politologii, biologii, sociologii, dopravě, IT a v mnoha jiných disciplínách. Hry se rozdělují podle různých kritérií:

- podle počtu hráčů - hry pro 2 hráče (šachy, dáma, ...) a hry pro více hráčů (poker, příklady z ekonomie, ...).
- podle náhody - rozdělení na hry deterministické (bez náhody - šachy, dáma, ...) a stochastické (do hry vstupuje náhoda - poker, prší, ...). Náhodu v teorii her označujeme pod pojmem "příroda". Jedná se o pseudo hráče, který zastupuje náhodu a nemá žádný cíl.
- podle informace - hry s úplnou a neúplnou informací (alespoň 1 z hráčů nezná část pravidel). Ve hrách s úplnou informací již oba hráči znají pravidla a tyto hry se dále dělí na hry s dokonalou informací (šachy - každý hráč zná po celou dobu stav, ve kterém se hra nachází a zároveň zná strategie své i svého protihráče) a hry s nedokonalou informací (poker - hráč nezná stav, ve kterém se hra aktuálně nachází. V tomto případě mu chybí informace ohledně soupeřových karet).
- podle zisku - hry s nulovým součtem (celkový zisk všech hráčů je roven nule - Go, šachy, poker) a hry s nenulovým součtem (hry většinou z reálného světa, kde celkový zisk všech hráčů je různý od nuly - Vězňovo dilema, Bitva pohlaví, v ekonomice).

2.1 Hry s nulovým součtem

Hry s nulovým součtem patří do kategorie her popisujících antagonistické konflikty. Zájmy hráčů jsou protichůdné a výše výhry jednoho hráče je rovna výši prohry druhého hráče. Nemá tedy smysl spolupracovat. Příkladem může být hra kámen, nůžky, papír. Tato práce se ale nezabývá hrami s nulovým součtem. O hrách s nulovým součtem se můžete dočíst v diplomové práci kolegy Martina Hurába [1], se kterým jsme společně pracovali na vývoji serveru.

2.2 Hry s nenulovým součtem

V diplomové práci se zabývám hrami s nenulovým součtem. Nejvíce příkladů těchto her nalezneme v ekonomice. Může se jednat o konkurenční boj dvou výrobců stejného zboží, o smlouvání mezi kupujícím a prodejcem, apod.

Hry mohou být definovány pomocí strategického nebo rozvinutého tvaru.

2.2.1 Strategická forma hry

Hry s nenulovým součtem se také nazývají hrami dvojmaticovými. Dvojmaticovou hrou rozumíme hru s konečnou množinou strategií R prvního hráče:

$$R = \{r_1, r_2, \ldots, r_m\}$$

a konečnou množinou strategií S druhého hráče:

$$S = \{s_1, s_2, \ldots, s_n\}$$

Při volbě strategií (r_i, s_j) jsou výhry hráčů:

$$a_{ij} = u_1(r_i, s_j)$$

$$b_{ij} = u_2(r_i, s_j)$$

kde u_1 a u_2 se nazývají výplatními funkcemi.

Hra se může reprezentovat jako bimatice, kde první číslo představuje výplatní funkci prvního hráče a druhé číslo představuje výplatní funkci druhého hráče. Bimatice má tolik řádků, kolik má první hráč různých strategií a tolik sloupců kolik má druhý hráč různých strategií. Následuje příklad bimatice:

$$\begin{pmatrix} (-3,1) & (5,2) & (4,7) & (10,-10) \\ (-1,2) & (4,3) & (6,1) & (1,8) \\ (0,12) & (1,2) & (2,2) & (9,3) \end{pmatrix}$$

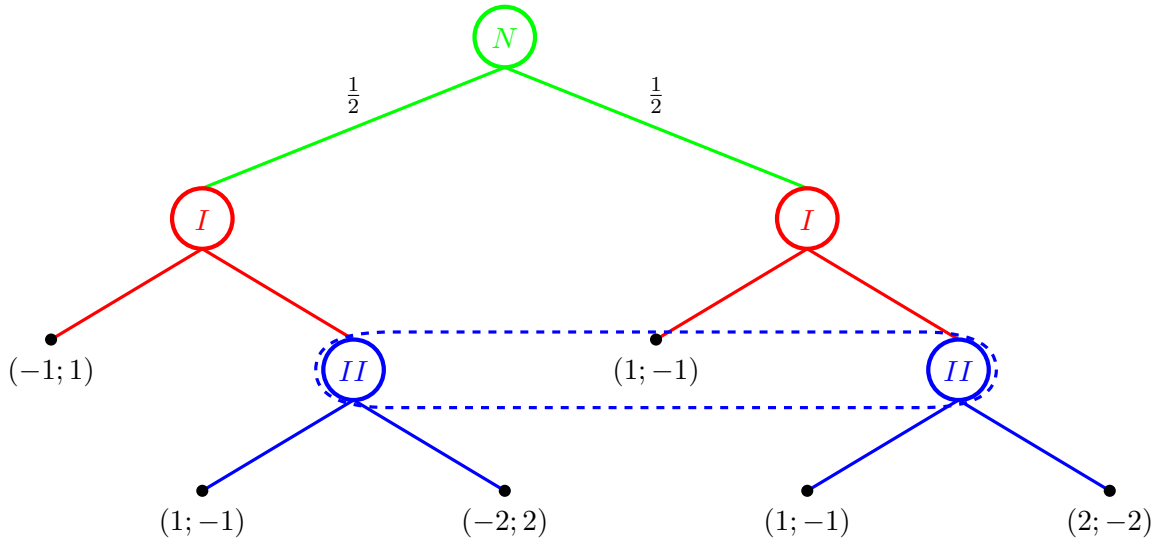
Z bimatice můžeme vidět, že první hráč má tři strategie a druhý hráč má strategie čtyři. Pokud první hráč zvolí druhou strategii a druhý hráč zvolí čtvrtou strategii, pak první hráč obdrží výplatu 1 a druhý hráč obdrží výplatu 8. Hru je také možno reprezentovat pomocí dvou matic.

$$A = \begin{pmatrix} -3 & 5 & 4 & 10 \\ -1 & 4 & 6 & 1 \\ 0 & 1 & 2 & 9 \end{pmatrix} B = \begin{pmatrix} 1 & 2 & 7 & -10 \\ 2 & 3 & 1 & 8 \\ 12 & 2 & 2 & 3 \end{pmatrix}$$

2.2.2 Rozvinutá forma hry

Rozvinutá forma hry je reprezentována pomocí stromové struktury, která se v teorii her označuje "Kuhnův strom". Strom hry zachycuje všechny situace, které mohou nastat. Každé situaci odpovídá jeden uzel a z každého uzlu vychází určitý počet hran, který odpovídá možným rozhodnutím, tzv. tahům hráče. Listy stromu pak reprezentují výplatní funkci hry a jsou zadány jako dvojice čísel. Na následujícím obrázku je ukázka Kuhnova stromu pro hru jednoduchý poker.

Hra v této formě dále obsahuje informační množiny, které určují množství informací, které bude mít daný hráč k dispozici. Ve hře typu šachy nebo dáma každý uzel stromu spadá do jiné informační množiny. Tyto hry jsou s dokonalou informací. Ve hře typu poker



Obrázek 1: Kuhnův strom

jsou už uzly, které spadají do stejných informačních množin. V tomto případě můžeme správným použitím informačních množin vystihnout to, že hráč nevidí karty soupeře.

V případě, že se jedná o stochastické hry, ještě hra v rozvinutém stavu obsahuje pseudohráče Příroda. U hran uzlu, kde probíhá náhodný tah, se nachází pravděpodobnosti. Součet pravděpodobností všech hran u daného uzlu musí být roven 1. Hráč Příroda odpovídá například zamícháním karet v karetních hrách nebo hodu kostkou ve hrách typu Backgammon nebo Člověče, nezlob se!

2.2.3 Redukce rozvinuté (extenzivní) formy hry na strategickou (normální)

Každou hru je možné zredukovat z extenzivní formy na strategickou. Na obrázku č. 1 můžeme vidět hru v rozvinutém tvaru. Převod na normální formu hry je následující:

- 1) Spočítáme počet strategií hráčů - celkový počet strategií daného hráče je dán jako násobek počtu potomků jednotlivých informačních množin daného hráče. V příkladu výše patří hráč I. do dvou různých informačních množin s 2 potomky. Celkový počet strategií I. hráče je tedy určen jako jejich násobek a je roven číslu 4. Hráč II. má všechny své strategie v jedné informační množině s 2 potomky. Počet strategií hráče II. je tedy roven 2.

- 2) Dopočítáme jednotlivé prvky matice. Řádky matice jsou určeny jako postupná kombinace jednotlivých informačních množin I. hráče a sloupce matice jsou určeny jako kombinace informačních množin II. hráče. Ukázka výpočtu prvního řádku:

$$A[1,1].X = (1:1) - (1:2) - (A:1) = \frac{1}{2} \cdot 2 + \frac{1}{2} \cdot -2 = 0$$

$$A[1,1].Y = (1:1) - (1:2) - (A:1) = \frac{1}{2} \cdot -2 + \frac{1}{2} \cdot 2 = 0$$

$$A[1,1] = (0,0)$$

$$A[1,2].X = (1:1) - (1:2) - (B:1) = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 1 = 1$$

$$A[1,2].Y = (1:1) - (1:2) - (B:1) = \frac{1}{2} \cdot -1 + \frac{1}{2} \cdot -1 = -1$$

$$A[1,2] = (1,-1)$$

- 3) Výsledná matice:

$$A = \begin{pmatrix} (0,0) & (1,-1) \\ (\frac{1}{2},-\frac{1}{2}) & (0,0) \\ (-\frac{1}{2},\frac{1}{2}) & (1,-1) \\ (0,0) & (0,0) \end{pmatrix}$$

2.2.4 Dominance

Jedná se o proces, který umožňuje zjednodušit hledání optimální strategie hry. Z matice se vyloučí všechny takové strategie, které jsou vždy horší než jiná strategie nebo kombinace jiných strategií. Dojde tím ke snížení rozměru matice a je poté snažší nalézt optimální strategii.

Strategie r_x hráče I. slabě dominuje strategii r_y téhož hráče, jestliže pro $j = 1, 2, ..., n$ platí

$$a_{yj} \leq a_{xj}$$

Strategie r_x hráče I. silně dominuje strategii r_y téhož hráče, jestliže pro $j = 1, 2, ..., n$ platí

$$a_{yj} < a_{xj}$$

Strategie s_x hráče II. slabě dominuje strategii s_y téhož hráče, jestliže pro $i = 1, 2, ..., m$ platí

$$b_{iy} \leq b_{ix}$$

Strategie s_x hráče II. silně dominuje strategii s_y téhož hráče, jestliže pro $i = 1, 2, ..., m$ platí

$$b_{iy} < b_{ix}$$

Užitím slabé dominance můžeme přijít o některé Nashovy rovnovážné body (viz. sekce 2.2.5) a tím bychom mohli přijít o optimální řešení hry. Z toho důvodu budeme nadále pracovat pouze se silnou dominancí, která nám redukuje rozměr matice a přitom zachovává optimální řešení hry. Příklad dominance:

$$A = \begin{pmatrix} (1,0) & (1,2) & (0,1) \\ (0,3) & (0,1) & (2,0) \end{pmatrix}$$

Z matice můžeme vidět, že sloupec 3 (strategie s_3) je dominován strategií s_2. Jedná se o silnou dominanci, protože jsme využili k porovnání ostrá znaménka: $1 < 2$ a $0 < 1$. Po eliminaci strategie s_3 dostaneme následující redukovanou matici:

$$A = \begin{pmatrix} (1,0) & (1,2) \\ (0,3) & (0,1) \end{pmatrix}$$

Z matice je patrné, že strategie s_1 nedominuje strategii s_2 a strategie s_2 nedominuje strategii s_1. Zaměříme se tedy na řádky matice (strategie I. hráče) a hned můžeme vidět, že strategie r_2 je silně dominována strategií r_1. Eliminujeme tedy druhý řádek.

$$A = \begin{pmatrix} (1,0) & (1,2) \end{pmatrix}$$

V posledním kroku výpočtu je strategie s_1 dominována strategií s_2. Po eliminaci prvního sloupce dostaneme redukovanou matici o 1 prvku, která představuje řešení hry s výplatami $(1,2)$.

$$A = \begin{pmatrix} (1,2) \end{pmatrix}$$

Dále je možné strategie dominovat pomocí kombinace jiných strategií. Takovouto dominanci si ukážeme na následujícím příkladu:

$$A = \begin{pmatrix} (1,1) & (1,2) & (0,4) \\ (0,3) & (0,1) & (2,0) \end{pmatrix}$$

V tomto příkladu již není možné dominovat jeden sloupec jiným sloupcem ani jeden řádek jiným řádkem. Je třeba využít kombinace sloupců. V tomto případě je možné dominovat strategii s_2 kombinací strategií s_1 a s_3 s pravděpodobností $\frac{1}{2}$. Kombinací strategií s_1 a s_3 na prvním řádkům získáme hodnotu $\frac{5}{2}$. Na druhém řádku pak získáme hodnotu $\frac{3}{2}$. Jedná se o silnou dominanci, protože jsme využili k porovnání ostrá znaménka: $2 < \frac{5}{2}$ a $1 < \frac{3}{2}$. Po eliminaci strategie $s2$ již získáme konečnou matici, kterou dále není možné pomocí dominance upravit a je třeba z ní dopočítat optimální strategii.

$$A = \begin{pmatrix} (1,1) & (0,4) \\ (0,3) & (2,0) \end{pmatrix}$$

Na rozdíl od her s nulovým součtem, kde nemá význam, aby hráči spolupracovali, je spolupráce hráčů u her s nenulovým součtem možná. Rozdělení her s nenulovým součtem je tedy následující:

2.2.5 Nekooperativní hry

Hráči spolu nespolupracují. Zde je stejně jako u her s nulovým součtem snaha o nalezení optimální strategie hráčů. V tomto typu her hledáme Nashův rovnovážný bod, který je reprezentován dvojicí rovnovážných strategií. Dále se u těchto her určuje: optimální strategie, ekvalizující strategie, bezpečná úroveň, a další.

Optimální strategie se může řešit tak, že si hru rozdělíme na 2 matice A a B^T a spočítáme jejich smíšené strategie pomocí metod od her s nulovým součtem. Více informací o hledání optimálních strategií se nachází v diplomové práci kolegy Martina Hurába. [1]

Bezpečná úroveň hry je částka, kterou může každý hráč zaručit, že ji v průměru vyhraje. Pro hru s maticemi A a B o rozměrech $m \times n$ Hráč I. může v průměru zaručit výhru nejméně:

$$v_I = \max_p \min_j \sum_{i=1}^{m} p_i \cdot a_{ij} = Val(A)$$

Bezpečná úroveň pro hráče II. je následující:

$$v_{II} = \max_q \min_i \sum_{i=1}^{n} b_{ij} \cdot q_j = Val(B)$$

Nalezení bezpečné úrovně si ukážeme na následujícím příkladu:

$$X = \begin{pmatrix} (2,0) & (1,3) \\ (0,1) & (3,2) \end{pmatrix}$$

Zadanou bimatici si můžeme rozdělit na 2 matice A a B pro každého hráče zvlášť:

$$A = \begin{pmatrix} 2 & 1 \\ 0 & 3 \end{pmatrix} B = \begin{pmatrix} 0 & 3 \\ 1 & 2 \end{pmatrix}$$

Z definic můžeme vidět, že bezpečná úroveň se vypočítá pomocí ceny hry, která je pro matici o rozměrech 2×2 určena pomocí následujícího vzorce z her s nulovým součtem:

$$Val(A) = \frac{a \cdot c - b \cdot d}{a - b + c - d}$$

Bezpečná úroveň pro prvního hráče se tedy spočítá následovně:

$$v_I = \frac{2 \cdot 3 - 1 \cdot 0}{2 - 1 + 3 - 0} = \frac{6}{4} = \frac{3}{2}$$

U druhého hráče je strategie s_1 dominována strategií s_2. Po eliminaci prvního sloupce získáme cenu hry:

$$v_{II} = 2$$

Ekvalizující strategie je strategie, která má stejnou bezpečnou úroveň bez ohledu na to, co dělá soupeř. Ekvalizující strategii nalezneme tak, že ji hráč spočítá ze soupeřovy matice. Ukázku výpočtu ekvalizující strategie si ukážeme na následujícím příkladu:

$$X = \begin{pmatrix} (3,3) & (0,2) \\ (2,1) & (5,5) \end{pmatrix}$$

Bimatici X si rozdělíme na 2 matice A a B:

$$A = \begin{pmatrix} 3 & 0 \\ 2 & 5 \end{pmatrix} B = \begin{pmatrix} 3 & 2 \\ 1 & 5 \end{pmatrix}$$

Pro matici A a B vyjdou následující optimální strategie:

$$p_A = (\frac{1}{2}, \frac{1}{2}) \quad q_A = (\frac{5}{6}, \frac{1}{6})$$
$$p_B = (\frac{4}{5}, \frac{1}{5}) \quad q_B = (\frac{3}{5}, \frac{2}{5})$$

Z toho vyplývají ekvalizující strategie pro matici X:

$$p = (\frac{4}{5}, \frac{1}{5}) \quad q = (\frac{5}{6}, \frac{1}{6})$$

Nashova rovnováha je situace, kdy žádný z hráčů nemůže změnou své strategie vylepšit svůj zisk. Své jméno tato situace dostala po americkém matematikovi a nositeli Nobelovy ceny za ekonomii v oblasti teorie her Johnu Forbesi Nashovi, který dokázal, že každá konečná hra má alespoň jedno takové řešení.

Podobně jako u dominance se i zde dělí Nashovy rovnovážné body na silné a slabé. Nashovým slabě rovnovážným bodem hry je dvojice strategií $(r*, s*)$ pro kterou platí:

$$(\forall r \in A)[A(r, s*) \leq A(r*, s*)] \quad \wedge \quad (\forall s \in B)[B(r*, s) \leq B(r*, s*)]$$

Nashovým silně rovnovážným bodem je dvojice strategií $(r*, s*)$ pro kterou platí:

$$(\forall r \in A \backslash r*)[A(r, s*) < A(r*, s*)] \quad \wedge \quad (\forall s \in B \backslash s*)[B(r*, s) < B(r*, s*)]$$

Ve hře dvou hráčů v oboru smíšených strategií je možné najít Nashovy rovnovážné body pomocí následující metody. Vše si ukážeme na příkladu:

$$A = \begin{pmatrix} (82, 82) & (125, 93) & (112, 78) \\ (93, 125) & (61, 61) & (84, 83) \\ (78, 112) & (83, 84) & (100, 100) \end{pmatrix}$$

Prvním krokem je nalezení sloupcových maxim pro výplaty prvního hráče a řádkových maxim pro výplaty druhého hráče. Nalezená maxima si označíme stříškou.

$$A = \begin{pmatrix} (82, 82) & (\widehat{125}, \widehat{93}) & (\widehat{112}, 78) \\ (\widehat{93}, \widehat{125}) & (61, 61) & (84, 83) \\ (78, \widehat{112}) & (83, 84) & (100, 100) \end{pmatrix}$$

Bod, ve kterém se nachází obě maxima je Nashovým rovnovážným bodem. V našem příkladu se jedná o dvě dvojice strategií (r_1, s_2) a (r_2, s_1). Nashovy rovnovážné body jsou následující:

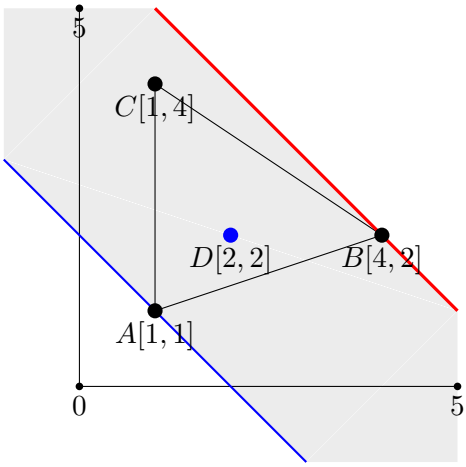
$$N = \{(r_1, s_2), (r_2, s_1)\}$$

Že se skutečně jedná o Nashovy rovnovážné body si můžeme jednoduše ověřit. Pokud první hráč volí strategii r_1 a druhý hráč se rozhodne změnit svou strategii na s_1 nebo s_3, pak se dostane do ztráty. Stejně tak i změna strategie prvního hráče z Nashovy rovnováhy nevede k lepšímu zisku.

2.2.6 Kooperativní hry s přenosnou výhrou

Kooperace spočívá v uzavření závazných dohod před realizací hry. Pro hry s přenosnou výhrou se hráči dohodnou na strategii a na přerozdělení výher. Vzhledem k tomu, že je možné celkovou výhru přerozdělit mezi oba hráče je možné provést rozdíl obou výplatních matic a zjednodušit si tak výpočet.

Množina dosažitelných řešení je u her s přenosnou výhrou šedá oblast, která se nachází mezi přímkami na obrázku:



Obrázek výše je vykreslen pro následující bimatici, kterou budeme používat jako ukázkový příklad pro hry s přenosnou strategiíi:

$$X = \begin{pmatrix} (1,1) & (4,2) \\ (1,4) & (2,2) \end{pmatrix}$$

Množina dosažitelných řešení v kooperativnách hrách s přenosnou výhrou je definována jako konvexní obálka k množině vektorů ve formě $(a_{ij} + s, b_{ij} - s)$, kde $i \in (1, \ldots, m)$, $j \in (1, \ldots, n)$ a s je libovolné reálné číslo.

Paretovo optimum Výsledek hry je Paretovým optimem pokud žádný hráč nemůže dosáhnout lepšího výsledku bez toho, že by se výsledek nějakého jiného hráče zhoršil. Na obrázku výše je Paretovým optimem červená přímka procházející bodem $A[4, 2]$.

Kooperativní strategie Pokud se hráči dohodnout na spolupráci, pak se budou snažit získat co největší zisk. Hledáme tedy co největší číslo po odečtení obou výplatních matic. Můžeme si ukázat volbu kooperativní strategie. Nejdříve převedeme bimatici X na matici Z jako součet obou výplatních matic:

$$Z = \begin{pmatrix} 2 & \widehat{6} \\ 5 & 4 \end{pmatrix}$$

V tomto případě nám vyšlo, že největší číslo je 6 a kooperativní strategií hry je volba strategií (r_1, s_2) s celkovým ziskem $\sigma = 6$.

Platba bokem Hráči se dále musí dohodnout na konečném rozdělení výher, kterým je vektor (x^*, y^*). Přitom platí, že $x^* + y^* = \sigma$. Pokud je zisk prvního hráče a_{ij} větší než x^*, pak musí provést platbu bokem druhému hráči a dát mu ze svého zisku částku $x^* - a_{ij}$. Pokud je ale zisk druhého hráče b_{ij} větší než y^*, pak musí druhý hráč provést platbu bokem a dát prvnímu hráči částku $y^* - b_{ij}$. Pokud by se v našem příkladu hráči dohodli na spravedlivém rozdělení výher, tak že $x^* = 3$ a $y^* = 3$, pak by musel první hráč provést platbu bokem o hodnotě 1 druhému hráči.

Strategie hrozby Hráči si dále volí své strategie hrozby. Pro nalezení strategií hrozeb je nejdříve potřeba provést rozdíl obou výplatních matic. Z bimatice X tak získáme matici W.

$$W = \begin{pmatrix} (1-1) & (4-2) \\ (1-4) & (2-2) \end{pmatrix} = \begin{pmatrix} 0 & 2 \\ -3 & 0 \end{pmatrix}$$

Strategie hrozby prvního hráče je strategie p z matice W a strategie hrozby druhého hráče je strategie q z matice W. Strategie hrozby pro oba hráče nám vyšly:

$$p = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad q = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Cena hry Z matice W můžeme dále spočítat cenu hry δ. V našem příkladu vychází cena hry $\delta = 0$.

Bod nedohody Když máme určeny strategie hrozeb, tak můžeme přejít k určení bodu nedohody. Nejprve je třeba rozdělit bimatici X na matice A a B.

$$A = \begin{pmatrix} 1 & 4 \\ 1 & 2 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 2 \\ 4 & 2 \end{pmatrix}$$

Pokud by nebyla dodržena dohoda, pak by první hráč získal $p^T \cdot A \cdot q$ a druhý hráč by získal $p^T \cdot B \cdot q$. Výsledný výplatní vektor se nazývá bodem nedohody D:

$$D = D(p, q) = (p^T \cdot A \cdot q, p^T \cdot B \cdot q) = (D_1, D_2)$$

V našem příkladu spočítáme bod nedohody následovně:

$$D_1 = (1 \quad 0) \cdot \begin{pmatrix} 1 & 4 \\ 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = (1 \quad 4) \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = 1$$

$$D_2 = (1 \quad 0) \cdot \begin{pmatrix} 1 & 2 \\ 4 & 2 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = (1 \quad 2) \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = 1$$

$$D = (1, 1)$$

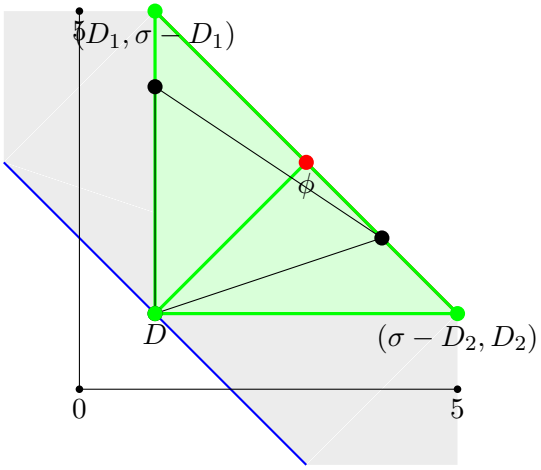
Rozdělení výher Když je určen bod nedohody, pak se musí oba hráči dohodnout na rozdělení výher (x, y), tak aby platilo, že $x+y = \sigma$. První hráč nepřijme zisk menší než D_1 a stejně tak druhý hráč nepřijme zisk menší než D_2. Přirozeným kompromisem je vybrat střední bod úsečky $|D_1D_2|$, který je určen následovně:

$$\phi = (\phi_1, \phi_2) = \left(\frac{\sigma - D_2 + D_1}{2}, \frac{\sigma - D_1 + D_2}{2} \right)$$

Výpočet si vyzkoušíme:

$$\phi_1 = \frac{6 - 1 + 1}{2} = 3$$
$$\phi_2 = \frac{6 - 1 + 1}{2} = 3$$
$$\phi = (\phi_1, \phi_2) = (3, 3)$$

Situaci můžeme řešit i graficky:



Bod ϕ lze tedy určit i z obrázku a to jako průsečík přímky z bodu D pod úhlem $45°$ s přímkou, která určuje Paretovo optimum.

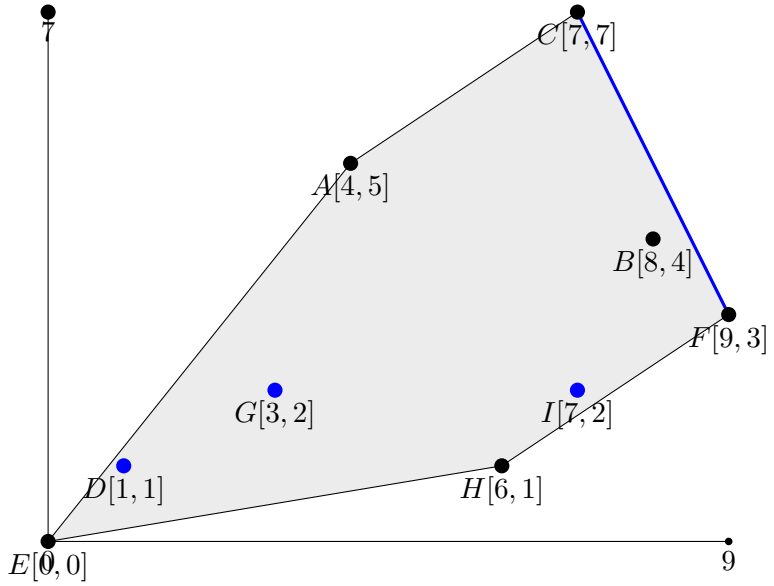
2.2.7 Kooperativní hry s nepřenosnou výhrou

V těchto hrách není možné přerozdělit výhry, hráči se pouze dohodnou na strategii, která je pro oba hráče výhodnější, než kdyby hráli samostatně.

Množina dosažitelných řešení Také hry s nepřenosnou výhrou mají množinu dosažitelných řešení. Ta je ale oproti hře s přenosnou výhrou určena konvexním obalem okolo množiny vektorů ve formě (a_{ij}, b_{ij}), kde $i \in (1, \ldots, m)$ a $j \in (1, \ldots, n)$. U her s nepřenosnou výhrou budeme pracovat s následujícím příkladem:

$$X = \begin{pmatrix} (4,5) & (8,4) & (7,7) \\ (1,1) & (0,0) & (9,3) \\ (3,2) & (6,1) & (7,2) \end{pmatrix}$$

Následuje grafické zobrazení množiny dosažitelných řešení kooperativních her s nepřenosnou výhrou:



K určení optimální výhry je třeba nalézt množinu nedominovaných dosažitelných výher a určit její střed. Optimální výhra je poté určena pomocí euklidovské vzdálenosti strategií od určeného středu množiny.

Paretovo optimum U nepřenosných výher jsou Paretovým optimem úsečky vytvořené spojením hraničních bodů konvexní obálky, které jsou vpravo nahoře. V našem případě je Paretovým optimem úsečka spojující vektory $(7,7)$ a $(9,3)$.

Zaručená výhra Nejnižší možná garantovaná výhra při zvolení své maxminové strategie. Výpočet zaručené výhry se nachází v tabulce č. 1. Z tabulky vycházejí zaručené výhry:

$$(h^z, g^z) = (4,2)$$

Dosažitelné výhry Výhry, které hráči zajišťují alespoň zaručenou výhru. Jedná se o všechny výhry, které se nacházejí na řádku nebo sloupci se zaručenou výhrou. V tomto příkladu se jedná o následující množinu:

$$D = \{(4,5), (8,4), (7,7), (9,3), (7,2)\}$$

X	s_1	s_2	s_3	min
r_1	$4,5$	$8,4$	$7,7$	$\underline{4}$
r_2	$1,1$	$0,0$	$9,3$	0
r_3	$3,2$	$6,1$	$7,2$	3
min	1	0	$\underline{2}$	

Tabulka 1: Výpočet zaručené výhry

Střed množiny Pro určení středu množiny nejprve odstraníme dominované dosažitelné výhry. Po odstranění je množina nedominovaných dosažitelných výher následující:

$$N = \{(8,4),(7,7),(9,3)\}$$

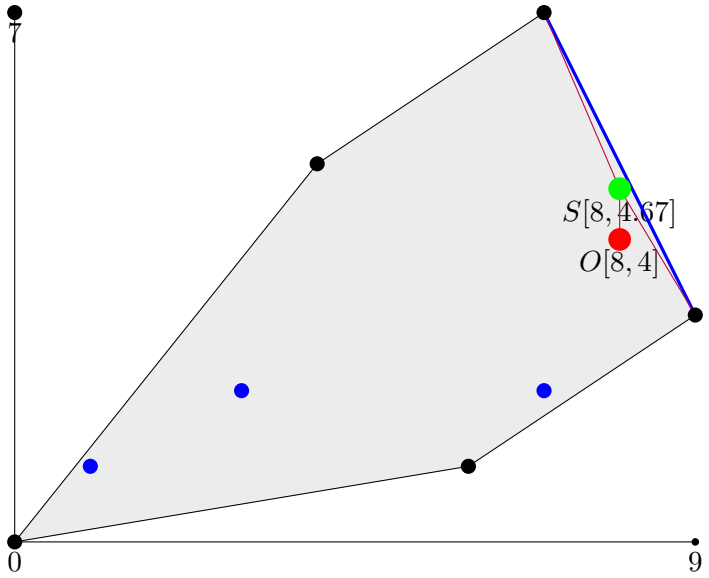
Střed množiny S poté spočítáme jako střední hodnotu všech výher množiny N.

$$S = \frac{((8,4)+(7,7)+(9,3))}{3} = (\frac{24}{3},\frac{14}{3}) \doteq (8,4.67)$$

Optimální výhra Jedná se o výhru, která má nejmenší euklidovskou vzdálenost od středu množiny nedominovaných dosažitelných výher. Zde můžeme rovnou vidět, že nejblíže k středu $S \doteq (8,4.67)$ má výhra $(8,4)$. Přesná vzdálenost v se určí pomocí klasického vzorce pro vzdálenost 2 bodů a a b v dvojrozměrném prostoru:

$$v = \sqrt{(a.x - b.x)^2 + (a.y - b.y)^2}$$

Výslednou situaci si zobrazíme graficky:



3 Analýza požadavků

Důležitou částí je analýza požadavků. Cílem bylo vytvoření webového serveru pro podporu výuky a bylo tedy nutné specifikovat funkční a nefunkční požadavky. Velká část požadavků byla určena již zadáním diplomové práce. Další požadavky přibývaly během konzultací s vedoucím diplomové práce.

3.1 Funkční požadavky

Funkčními požadavky se rozumí funkcionalita a chování aplikace. Základní otázky pro funčkní požadavky jsou: proč, k čemu, kdo, vstupy, výstupy a funkce.

3.1.1 Proč je zapotřebí nový systém?

Systém je vytvářen jako série diplomových prací, která slouží k podpoře výuky předmětu Teorie her. Minulý rok byl vytvořen server, který se týkal kombinatorických her. Nyní se pracuje na serveru, který je určen pro podporu výuky her s nulovým a nenulovým součtem. Důvodem k vytvoření nového systému je propojení všech tří částí do jedné webové aplikace, která bude uživatelsky přívětivá pro studenty a jejím cílem bude usnadnit jim pochopit probíranou látku a prakticky si vyzkoušet hry.

3.1.2 K čemu má systém sloužit?

Systém má sloužit k podpoře výuky teorie her. Hlavním cílem je ukázat na příkladech řešení her s nenulovým součtem pro nekooperativní hry, kooperativní hry s přenosnou výhrou a kooperativní hry s nepřenosnou výhrou. Dále bude systém sloužit k tomu, aby si studenti mohli vyzkoušet různé nenulové hry zahrát proti počítači.

3.1.3 Kdo bude se systémem pracovat?

Systém je určen hlavně pro studenty předmětu Teorie her. K užívání systému není třeba mít přístupové údaje, takže se systémem můžou pracovat i nejrůznější cizí uživatelé. Systém by měl proto mít jednoduché a intuitivní ovládání. Noví uživatelé se v systému zorientují pomocí nápovědy.

3.1.4 Jaké budou vstupy do systému?

Vstupy aplikace jsou následující:

Kuhnův strom: musí být ve stromové struktuře a musí se vytvářet dynamicky podle reakce na uživatelský vstup.

Bimatice: alternativa pro zadání her s nenulovým součtem. Vzhledem k různým velikostem her, je třeba vytvářet bimatici také dynamicky.

Další vstupy: nastavení obtížnosti, počet pokusů pro simulaci.

3.1.5 Jaké budou výstupy ze systému?

U všech typů her je výstupem bimatice a výplatní matice obou hráčů. Další výstupy si rozdělíme podle toho, v jaké části aplikace se nacházejí:

- **Nekooperativní hry** - Nashovy rovnovážné body, matice hráčů, řešení dominování, optimální strategie, bezpečná úroveň a ekvalizující strategie.
- **Kooperativní hry s přenosnou výhrou** - grafický náhled řešení, strategie hrozby, cena hry, kooperativní strategie, rozdělení výher, platba bokem, bod nedohody.
- **Kooperativní hry s nepřenosnou výhrou** - grafický náhled řešení, zaručená výhra, množina dosažitelných výher, množina nedominovaných dosažitelných výher, střed množiny, optimální výhra.
- **Ostatní výstupy** - hry proti počítači a simulace her a s tím související výstupy.

3.1.6 Jaké bude mít systém funkce?

Systém musí mít následující funkce: redukce Kuhnova stromu na bimatici, výpočet řešení včetně grafického zobrazení pro kooperativní i nekooperativní hry, hraní her proti počítači se zvolenou obtížností, simulace vybraných her, atd.

3.2 Nefunkční požadavky

Nefunkční požadavky kladou důraz na design aplikace, odezvu, bezpečnost, použitý programovací jazyk, na použité standardy, atd.

3.2.1 Design aplikace

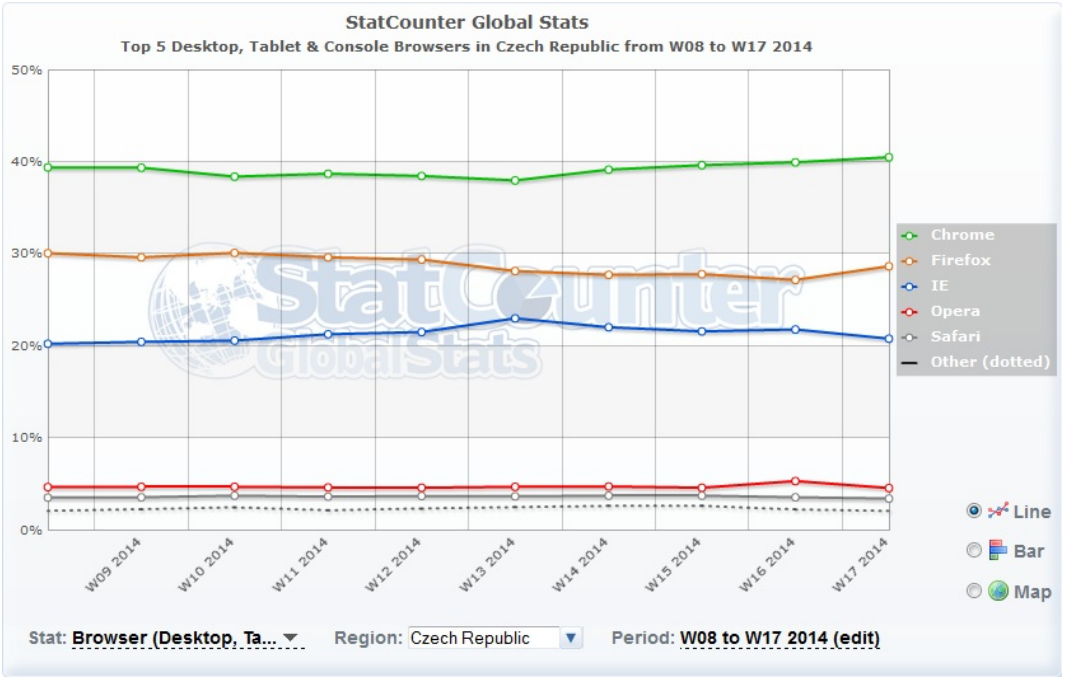
Je třeba, aby byla aplikace jednoduchá a intuitivní na používání. Aplikace je přehledně rozdělena na 3 části pomocí horního menu a spojuje tak celou sérii diplomových prací v rámci jedné webové stránky.

3.2.2 Použitá technologie

Server bude vyvíjen v prostředí Microsoft Visual Studio v programovacím jazyku C#. K reálnému nasazení je třeba server s podporou technologie ASP.NET.

3.2.3 Odezva aplikace

Rychlost odezvy je důležitým parametrem. Je třeba, aby uživatelé nemuseli dlouho čekat na výsledek výpočtu. Proto je důležité zvolit vhodné metody pro výpočet a omezit velikost vstupních dat na rozumnou hodnotu.



Obrázek 2: Statistiky používání internetových prohlížečů

3.2.4 Použité standardy

Jedná se o webovou aplikaci a stránky tedy musí být přístupné uživatelům. Je důležité se postarat, aby uživatelé mohli na stránky přistupovat z různých prohlížečů. Ke zjištění jak vypadají stránky v mnoha různých prohlížečích můžeme využít analýzu z následujících stránek: *https://browsershots.org/*. [2]

Vývoj podílů používání vyhledávačů je dynamický proces a statistiky se mění každým rokem. Aktuální situaci charakterizují statistiky společnosti StatCounter [3] na obrázku č. 2. Statistiky jsou zobrazeny pro 9. - 17. týden roku 2014 pro Českou republiku. Vyplývá z nich, že nejpoužívanější prohlížeče jsou u nás v tomto pořadí: Google Chrome, Mozilla FIrefox, Internet Explorer, Opera, Safari a další.

4 Návrh serveru

Následuje návrh struktury serveru a grafického rozhraní.

4.1 Architektura

Celý systém je programován pomocí technologie ASP.NET, která je součástí .Net Frameworku. Hlavními komponentami ASP.NET architektury jsou webové formuláře, code-behind soubory a kompilované DLL soubory. Webové formuláře obsahují HTML elementy, texty a serverové ovládací prvky. Code-behind soubory obsahují logiku aplikace pro webové formuláře. Kompilované DLL soubory poskytují dynamické HTML na webovém serveru. Náhled ASP.NET architektury se nachází na obrázku č. 3.

Společná Architektura systému pod technologií ASP.NET je následující [4]:

1. **Projekt UI** - obsahuje webové formuláře a slouží k zobrazení stránek.
2. **Projekt ZeroSum** - logika aplikace pro hry s nulovým součtem.
3. **Projekt NonZeroSum** - logika aplikace pro hry s nenulovým součtem.

4.2 Grafické uživatelské rozhraní

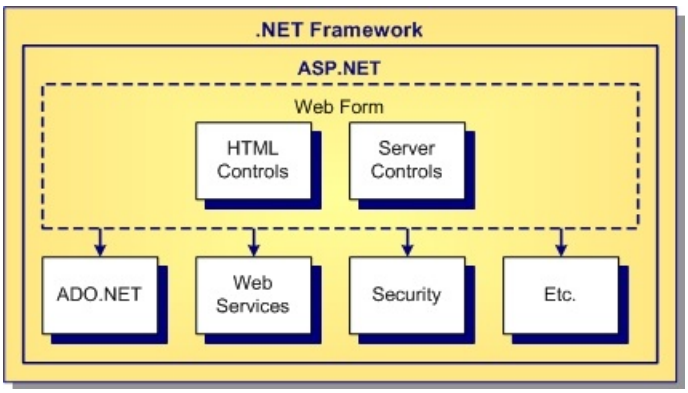
Následuje popis grafického rozhraní aplikace.

4.2.1 Hlavní menu

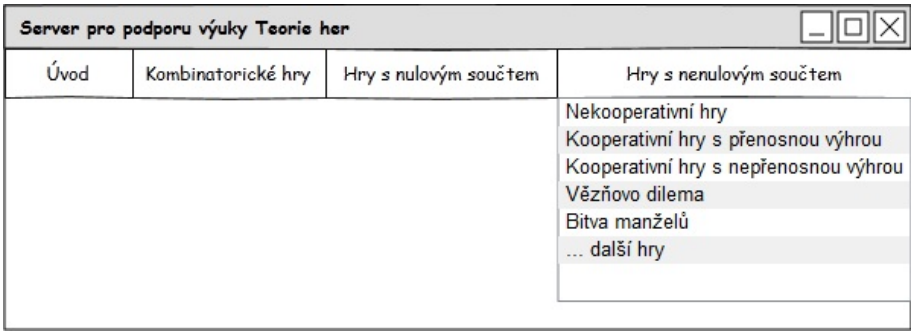
Webová aplikace má působit jako jednotný web k výuce Teorie her. Vzhledem k tomu, že se aplikace skládá ze 3 různých částí teorie her, je hlavní prioritou rozdělit aplikaci na 3 samostatné části. S kolegou Martinem Hurábem jsme se rozhodli, že aplikaci rozdělíme pomocí horního menu na následující části:

1. **Úvod** - stručný úvod k struktuře a obsahu systému.
2. **Kombinatorické hry** - část pro kombinatorické hry, která byla vytvořena před rokem jiným studentem jako první diplomová práce ze série. [5] Tato práce byla zpracována v programovacím jazyce Java za pomoci frameworku Vaadin. Z toho důvodu se zde nachází pouze hypertextový odkaz na adresu, kde je tento server spuštěn.
3. **Hry s nulovým součtem** - část pro hry s nulovým součtem vytvářena kolegou Martinem Hurábem.
4. **Hry s nenulovým součtem** - část pro hry s nenulovým součtem, která je součástí této diplomové práce.

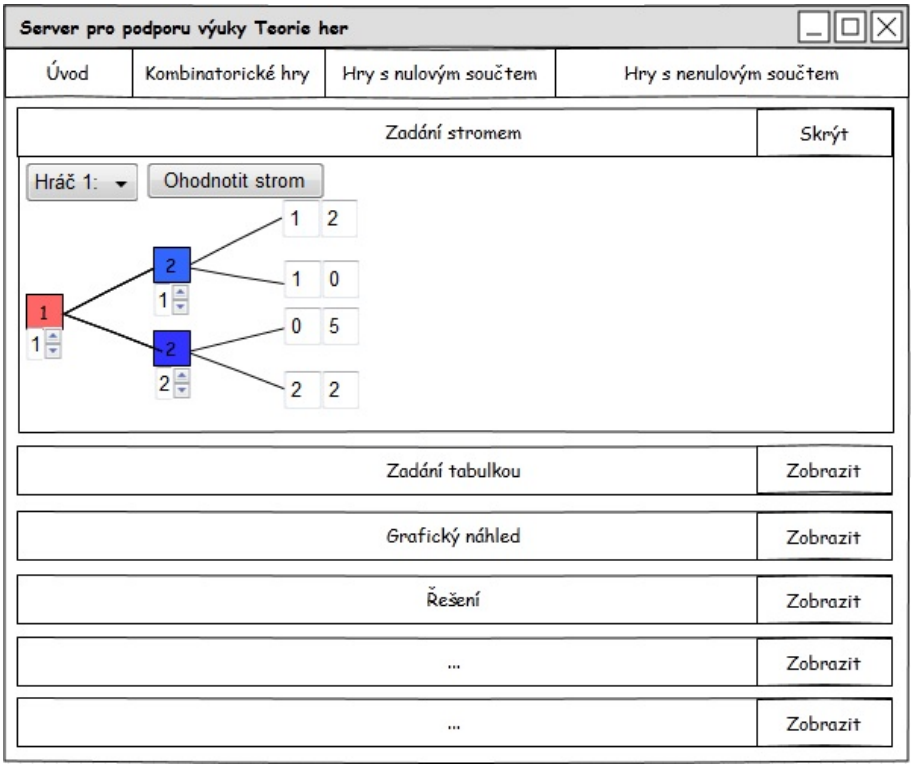
Následuje ukázka návrhu hlavního menu na obrázku č. 4. Všechny ukázky aplikací jsou vytvořeny ve volně přístupné aplikaci pro návrh GUI, která se jmenuje Pencil.



Obrázek 3: Architektura ASP.NET



Obrázek 4: Návrh hlavního menu



Obrázek 5: Návrh hlavních typů her s nenulovým součtem

4.2.2 Hlavní 3 typy her

Dále následuje návrh obrazovky pro zadání a řešení hlavních 3 typů her s nenulovým součtem. V aplikaci jsem se snažil o vytvoření jednotného vzhledu pro všechny typy her podle návrhu na obrázku č. 5.

Vzhledem k velkému počtu dat k výpisu jsou tato data rozdělena do jednotlivých sekcí - rozevíracích panelů. Dvěmi základními sekcemi jsou:

- **Zadání stromem** - první variantou zadání je zadat příklad pomocí Kuhnova stromu. Tvorba stromu probíhá v následujících fázích:

  1. **Tvorba stromu** - v první části si definujeme strukturu Kuhnova stromu. Tvorbu stromu můžeme dále rozdělit na následující části:
     - **přidání nového uzlu** - pokud chceme do stromu přidat nový uzel, pak je nejdříve třeba vybrat z rozbalovací nabídky, zda vytváříme uzel pro prvního hráče, druhého hráče a nebo pro náhodu. Když máme vybráno, tak můžeme kliknout na fialové tlačítko "+", které se nachází na listech stromu pro vytvoření nového uzlu. Tímto způsobem se nám vždy vytvoří rovnou 2 nové uzly.

  - **přidání potomka** - pokud chceme nějakému uzlu přidat více potomků, pak stačí kliknout na dané tlačítko rodiče. Potomek bude stejného typu (Hráč 1, Hráč2, Náhoda) jako jsou další potomci jeho rodiče.
  - **smazání potomka** - potomka můžeme smazat kliknutím na červené tlačítko "-"na listech stromu.

  Jakmile jsme spokojeni s aktuální strukturou stromu, tak můžeme přejít na ohodnocení stromu kliknutím na tlačítko "Ohodnotit strom". Tím se přepneme do další fáze.

2. **Ohodnocení stromu** - v této části přiřadíme Kuhnovu stromu informační množiny, pravděpodobnostní rozdělení a výplaty. Jednotlivé části si postupně rozebereme:
  - **nastavení informačních množin** - pod každým uzlem hráče si můžeme zvolit číslo od 1 do 4. Vybrané číslo nám bude určovat informační množinu, do které daný uzel od daného hráče spadá. Prvky hráče ve stejné informační množině musí mít stejný počet potomků. V případě porušení této podmínky se nám objeví informace, že je třeba opravit informační množiny.
  - **nastavení výplat hráčů** - na listech stromu se místo tlačítek "+"a "-"nachází dvě textová pole, do kterých je možné zadat výplatu pro oba hráče.
  - **nastavení pravděpodobností u náhody** - u uzlů reprezentujících náhodu se na každé výstupní hraně objevují 2 textová pole. Slouží k zadání čísla ve formě zlomku. Podmínkou je, že součet pravděpodobností daného uzlu musí být roven číslu 1. V případě porušení této podmínky se objeví chybová hláška.

  Stisknutím tlačítka "Vypočítat"dojde k převodu Kuhnova stromu do tabulky níže a rovnou i k výpočtu celkového řešení dané hry. Strom se pak přepne zpět do fáze 1 a je možné opět změnit jeho strukturu a následně jej znovu ohodnotit.

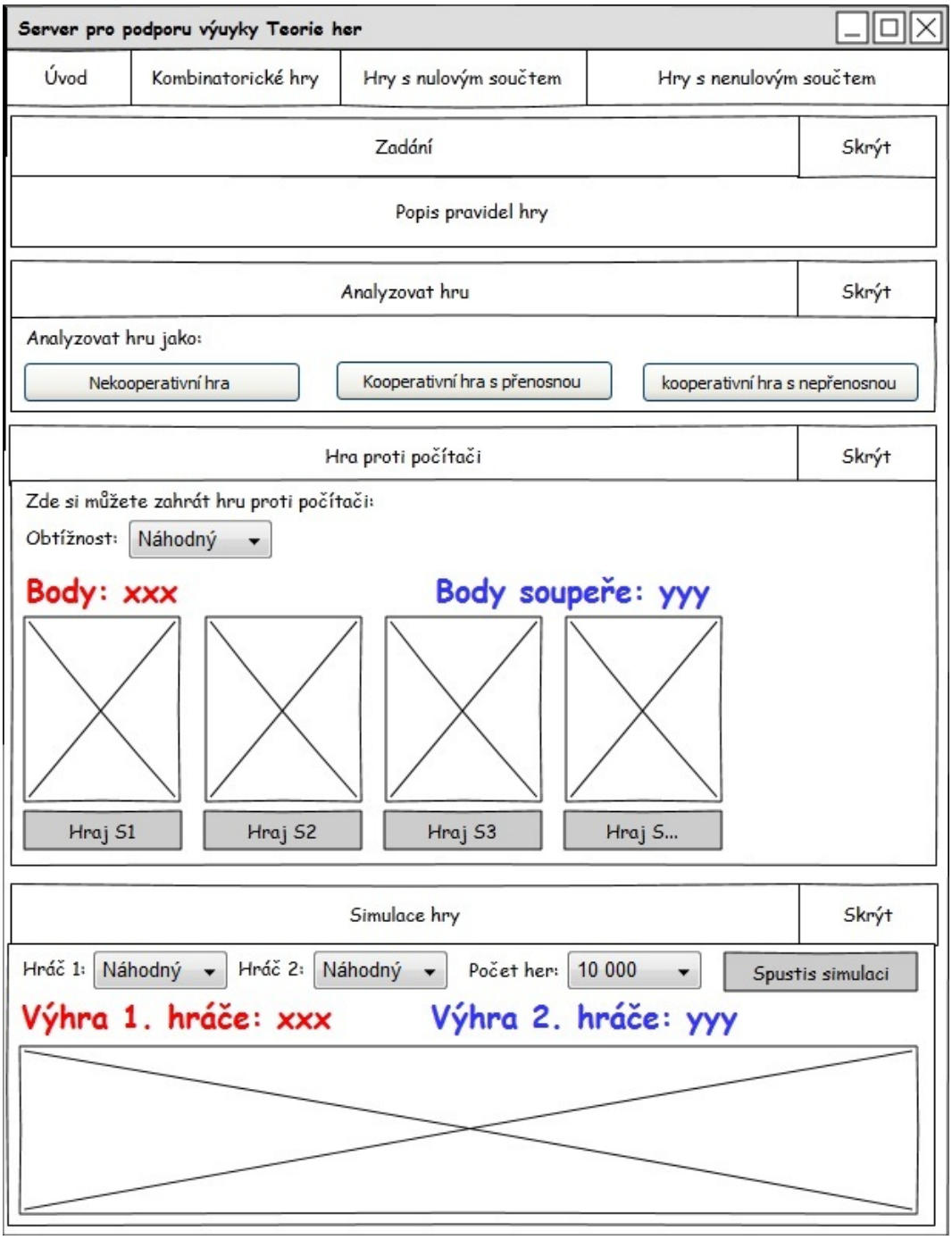
- **Zadání tabulkou** - hru je možné zadat tabulkou. Tabulku můžeme získat buď vyplněním Kuhnova stromu a nebo ručním zadáním. V případě ručního zadání je nejprve třeba vybrat počet řádků a sloupců z rozbalovací nabídky a následně vyplnit celou bimatici. Poté stačí stisknou tlačítko "Vypočítat"a provede se výpočet celkového řešení zadané hry.

Další sekce už se liší podle typu dané hry. Tyto sekce jsou rozděleny podle toho, co nás u dané hry zajímá. U kooperativních her je například zobrazen grafický náhled množiny dosažitelných řešení včetně řešení hry. U nekooperativních her máme sekci pro dominování strategií.

Na konci stránky pak mají všechny typy her sekci "Řešení", kde se nachází celkové řešení hry.

4.2.3 Hry proti počítači a simulace

V této části si ukážeme, podle jakého vzoru se zobrazují hry proti počítači a simulace her. Návrh se nachází na obrázku č. 6.



Obrázek 6: Návrh pro hru proti počítači a pro simulaci

Zadání První sekcí je zadání hry. V této sekci se hráč seznámí s pravidly hry a také se dozví, co je cílem hry proti počítači.

Analýza hry V této sekci se zadání vybrané hry pošle k zvolené analýze stisknutím tlačítka. Návrh analýzy hry je popsán v předchozí kapitole 4.2.2.

Hra proti počítači Zde je možné zahrát si hru proti počítači. Hráč si zvolí obtížnost soupeře z rozbalovací nabídky. Na výběr jsou následující obtížnosti: Náhodný, Začátečník, Pokročilý, Expert a Optimální. Na základě vybrané obtížnosti pak počítač volí, v jakém poměru bude hrát náhodnou a optimální strategii. Obtížnost je možné měnit i během hry. Poté stisknutím tlačítka "Nová hra"spustí hru. Hra spočívá ve výběru strategie s_1 až s_n pomocí tlačítek. Nad každým tlačítkem je zobrazen ilustrační obrázek. Na základě výběru hráče a následného tahu protihráče dostanou hráči body. Po každé volbě strategie se ukáže informační hláška, kolik bodů daný hráč obdržel a dále se objeví průběžné skóre obou hráčů. Hra končí po splnění podmínky pro konec hry, která je definována v zadání dané hry.

Simulace hry Poslední sekcí je simulace hry, ve které si uživatel zvolí obtížnost pro prvního a druhého hráče a počet odehraných her. Aplikace následně provede simulaci. Na základě této simulace zobrazí průměrnou výhru jednotlivých hráčů a také vykreslí průběžný graf pro oba hráče.

4.3 Uživatelská příručka

Vyhledem k tomu, že se jedná o server pro podporu výuky Teorie her, je nutné, aby studenti měli k dispozici uživatelskou příručku, pomocí které se seznámí se všemi typy her. Příručka je koncipována podle jednotlivých typů her s nenulovým součtem do jednotlivých kategorií.

5 Popis implementace

Implementace probíhala průběžně v iteracích na základě pravidelných konzultací s vedoucím diplomové práce a po dohodě s kolegou Martinem Hurábem na jednotném vzhledu a architektuře projektu.

5.1 Vývojové prostředí

Jako vývojové prostředí jsem si zvolil Microsoft Visual Studio 2010 v základní edici Professional. Visual studio může být použito pro vývoj desktopových i webových aplikací a také podporuje velké množství programovacích jazyků jako jsou: C, C++, C#, VB.NET... Doinstalováním dalších jazykových služeb je v podstatě možné programovat v libovolném jazyce. Projekt je implementován jako webová aplikace v programovacím jazyce C# pod technologií ASP.NET.

5.2 Logika aplikace

Logika aplikace se nachází v projektu NonZeroSum. Dále je zde projekt UI, který slouží k zobrazení GUI a volá si metody z aplikační logiky. Projekt UI je napojen na aplikační logiku her s nulovým součtem od kolegy Martina Hurába (ZeroSum) i na aplikační logiku her s nenulovým součtem (NonZeroSum). Projekt pro hry s nenulovým součtem obsahuje následující třídy:

- **Calculations** - třída, která obsahuje pomocné výpočty.
- **DrawImage** - třída, která se stará o vykreslení obrázků. Často jsou v této třídě vstupními parametry matice nebo Kuhnův strom a dále cesta pro uložení obrázku.
- **Leaf** - třída pro reprezentaci listu stromu.
- **NumberConverter** - třída pro převod čísel.
- **PlayerType** - výčtový typ pro hráče - Hráč 1 = 1, Hráč 2 = 2, Příroda = 99
- **PointReal** - třída pro reprezentaci dvourozměrného bodu v oboru reálných čísel.
- **Tree** - třída pro reprezentaci Kuhnova stromu.

5.3 Uživatelské rozhraní

Pro zobrazení uživatelského rozhraní slouží již zmíněný projekt UI. Systém má jednotný vzhled, takže stačilo vytvořit hlavní Master Page (Site.Master), která obsahuje společné prvky a na vyznačené místo umožňuje zadat obsah pro zvolenou stránku. V této Master Page je dále uloženo hlavní menu aplikace. Jsou na ní navázány CSS styly stránky (style.css) a rozšíření pro technologii ASP.NET AJAX, které se jmenuje ASP.NET AJAX Control Toolkit a slouží jako dobrý nástroj k psaní rozšiřujících interaktivních ovládacích prvků. [6]

Následuje struktura projektu UI. Vypsány jsou pouze třídy, které se týkají her s nenulovým součtem:

- **NonZeroSumUserGuide** - uživatelská příručka
- **NonZeroSumTheory** - stránka pro studenty obsahující teorii k hrám s nenulovým součtem
- **NonZeroSumNonCooperativeGames** - třída pro analýzu nekooperativních her
- **NonZeroSumCooperativeTUGames** - třída pro analýzu kooperativních her s přenosnou výhrou
- **NonZeroSumCooperativeNTUGames** - třída pro analýzu kooperativních her s nepřenosnou výhrou
- **NonZeroSumSimplePoker** - třída pro hru jednoduchý poker
- **NonZeroSumBattleOfTheSexes** - třída pro hru Bitva pohlaví
- **NonZeroSumPrisonersDilemma** - třída pro hru Vězňovo dilema
- **NonZeroSumTender** - třída pro hru Výběrové řízení
- **NonZeroSumDistributionOfLeaflets** - třída pro hru Distribuce letáků

5.4 Reprezentace dat

Dále je nutné zvolit vhodnou reprezentaci dat. Již v kapitole 2.2 jsme se mohli dočíst, že data u her s nenulovým součtem jsou zadána v podobě matice a nebo Kuhnova stromu. Tato kapitola tedy pojednává hlavně o tom, jakým způsobem jsou následující 2 formy her uloženy:

5.4.1 Rozvinutá forma hry

Pro reprezentaci stromové struktury jsem zvolil třídu NTree. Třída je definována následovně:

```
public class NTree
    {
        private const int MAXLEVEL = 8;
        private static  int  nodeId = 1;
        private static  int  node1Count = 0;
        private static  int  node2Count = 0;
        private int  id;
        private int  player;
        private int  level;
        private int  informationSet;
        private double random, randomNumerator, randomDenominator;
        private double x, y;
```

```
        LinkedList<NTree> children;

        public NTree(int player = 0, int level = 1, int informationSet = 1, double random = 0,
            double randomNumerator = 1, double randomDenominator = 2, double x = 0, double
            y = 0)
        {
            id = nodeId;
            nodeId++;
            this.player = player;
            this.level = level;
            this.informationSet = informationSet;
            this.random = random;
            this.randomNumerator = randomNumerator;
            this.randomDenominator = randomDenominator;
            this.x = x;
            this.y = y;
            children = new LinkedList<NTree>();
        }
    ...
}
```

Výpis 1: Reprezentace extenzivní formy hry

Kuhnův strom obsahuje jedinou konstantu MAXLEVEL, která má hodnotu 8 a určuje nám maximální hloubku stromu. Hodnota je omezena z důvodu grafického zobrazení a času na výpočet.

Strom dále obsahuje tři statické proměnné nodeId, nodeCount1, nodeCount2, které určují jednoznačný identifikátor uzlu stromu a počet uzlů daného hráče.

Dále strom obsahuje třídní proměnné id, player, level, informationSet, random, randomNumerator, randomDenominator, x, y a children. První čtyři proměnné určují id daného uzlu, číslo hráče, úroveň zanoření a informační množinu. Další tři proměnné slouží k uložení náhody u uzlů, které patří Přírodě. Poslední proměnná children je spojovým seznamem reprezentovaným generickou kolekcí LinkedList a obsahuje všechny potomky daného uzlu. V případě, že je tato proměnná prázdná, tak se jedná o list stromu.

Níže se nachází konstruktor třídy s předem nadefinovanými třídními parametry. Můžeme si všimnout, že se při tvorbě stromu inkrementuje statická proměnná nodeId, která nám určuje jednoznačný identifikátor daného stromu. Na konci metody se pak nachází inicializace potomků na prázdný seznam stromů.

Pod konstruktorem se pak místo tří teček nachází další metody.

Třída je pomocí potomků definována rekurzivně. Následuje ukázka kódu pro vytvoření hry Bitva pohlaví:

```
NTree.clearStaticVariables();
NTree tree = new NTree();
tree.addChild((int)PlayerType.PLAYER1);
tree.addChild((int)PlayerType.PLAYER1);
NTree bach1 = tree.getChild((int)PlayerType.PLAYER1);
NTree stravinskij1 = tree.getChild((int)PlayerType.PLAYER2);
bach1.addChild((int)PlayerType.PLAYER2);
bach1.addChild((int)PlayerType.PLAYER2);
stravinskij1 .addChild((int)PlayerType.PLAYER2);
```

```
 stravinskij1 .addChild((int)PlayerType.PLAYER2);
NTree bach2 = bach1.getChild((int)PlayerType.PLAYER1);
NTree stravinskij2 = bach1.getChild((int)PlayerType.PLAYER2);
NTree bach22 = stravinskij1.getChild((int)PlayerType.PLAYER1);
NTree stravinskij22 = stravinskij1 .getChild((int)PlayerType.PLAYER2);
bach2.setValue(2, 1);
 stravinskij2 .setValue(0, 0);
bach22.setValue(0, 0);
 stravinskij22 .setValue(1, 2);
```

Výpis 2: Hra Bitva pohlaví reprezentována Kuhnovým stromem

Na prvních 2 řádcích dochází k inicializaci statických proměnných a k vytvoření nového stromu. Na daný strom se 2x volá metoda addChild, která přidá 2 potomky prvního hráče. Pak se pomocí metody getChild uloží oba potomci do podstromů bach1 a stravinskij1. Tyto podstromy poté mají oba po dvou potomcích druhého hráče. Tím dostaneme 4 listy stromu, které se na posledních 4 řádcích odhodnotí pomocí metody setValue. Tímto stromem jsme zadali tuto hru určenou bimaticí:

$$X = \begin{pmatrix} (2,1) & (0,0) \\ (0,0) & (1,2) \end{pmatrix}$$

Třída NTree dále obsahuje spoustu metod využívajících rekurzi a průchod stromem. Více se o těchto metodách dočtete v dokumentaci.

Dále bylo třeba u stromu provést validaci vstupních proměnných. Kontoluje se, zda do stromu vstupují reálná čísla v intervalu $(-500, 500)$, zda mají stejné informační množiny stejný počet prvků a zda je součet pravděpodobností roven 1.

Ukázku stromu v aplikaci naleznete na obrázku č.7.

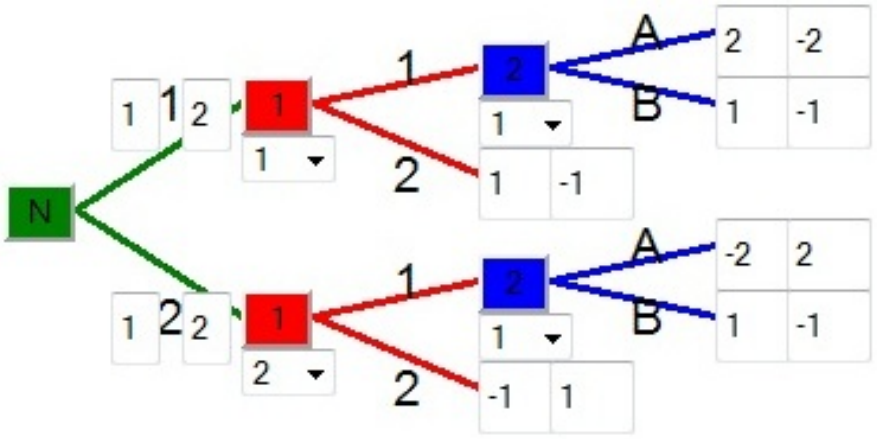
5.4.2 Strategická forma hry

Normální forma hry je bimatice. V programu je reprezentována pomocí dvojrozměrného pole typu PointReal. Následuje ukázka inicializace matice na počáteční hodnotu 0.

```
PointReal [][] matrix = new PointReal[colsCount][];
for (int i = 0; i < colsCount; i++)
{
    matrix[i] = new PointReal[rowsCount];
    for (int j = 0; j < rowsCount; j++)
    {
        matrix[i][j] = new PointReal();
        matrix[i][j].setX(0);
        matrix[i][j].setY(0);
    }
}
```

Výpis 3: Normální forma hry

Také zde bylo třeba validovat vstupní proměnné. Pro všechna pole bimatice se kontroluje, zda se jedná o reálná čísla z intervalu $(-500, 500)$.



Obrázek 7: Kuhnův strom v aplikaci

Na obrázku č. 8 následuje ukázka zobrazení bimatice v běžící aplikaci pro hru Bitva pohlaví. Ukázka je zobrazena včetně celkového vzhledu aplikace. Pro přehlednost byly všechny sekce, které se netýkají ukázky bimatice, skryty.

5.5 Použité algoritmy

Tato podkapitola se věnuje známým algoritmům, které byly použity při tvorbě aplikace.

5.5.1 Rekurze

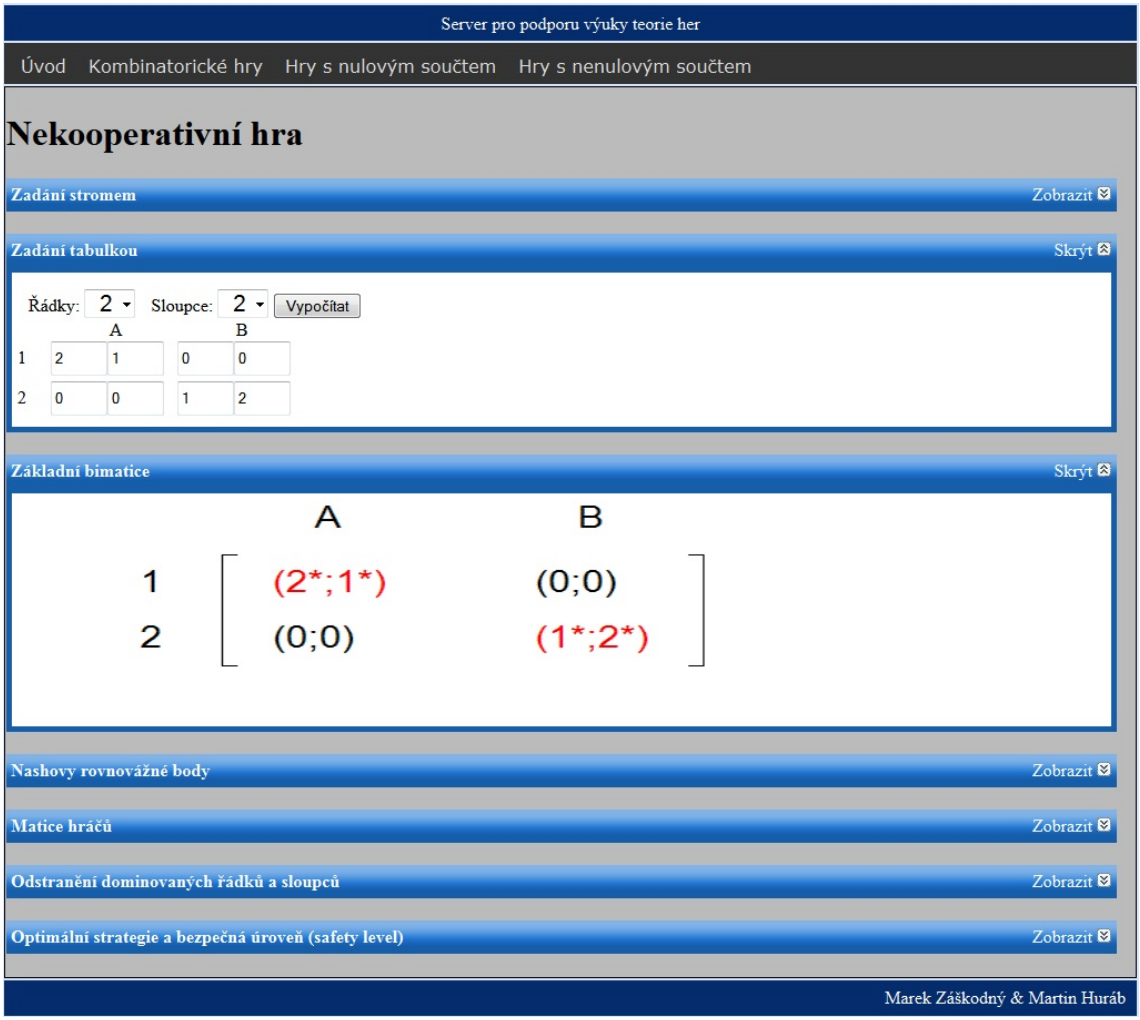
Rekurze je jedním z velmi často používaných algoritmů. Rekurzivní funkce je funkce, která volá sama sebe většinou s pozměněnými parametry, dokud není splněna ukončovací podmínka. V aplikaci je rekurze použita v třídě NTree, při vykreslování stromové struktury v projektu UI na jednotlivých stránkách. V následujícím kódu můžeme vidět kód metody, který slouží ke konstrukci stromu pro zadání hodnot:

```
void constructMainTree(NTree tree, string path, int x = 0, int y = 0)
{
    ... // vykresleni stromu
    constructMainTreeRecursion(tree, x, y);
    Image.ImageUrl = "/Temp/" + Session.LCID + ".gif";// zobrazeni stromu ve formulari
}
```

Výpis 4: Vyvolání rekurze

Tato metoda volá rekurzivní metodu constructMainTreeRecursion. Vzhledem k délce metody následuje jen zkrácená ukázka kódu:

```
void constructMainTreeRecursion(NTree tree, int x = 0, int y = 0, int width = 30, int
     height = 25, int gapBetween = 100)
{
    if (!tree.isLeaf()) // pokud se nejedna o list
```



Obrázek 8: Reprezentace bimatice v aplikaci

```
        {
            Button add = new Button();
            ... // nastaveni vlastnosti tlacitka vcetne polohy podle souradnic x a y
            add.Click += (sender, e) => { add_Click(sender, e, tree.getId()); };
            ... // pridani tlacitka a inicializace promennych
            foreach (var child in tree.getChildrens()) // pro vsechny potomky
            {
                ... // kontrola souctu nahodneho uzlu
                constructMainTreeRecursion(child, x, y + 30 * i); // VOLANI REKURZE
                ... // pocitani sirky stromu
            }
            ... // kontrola pravdepodobnosti
        }
        else
        {
            ... // v pripade ze se jedna o list tak se vykresli vcetne tlacitek
        }
    }
```

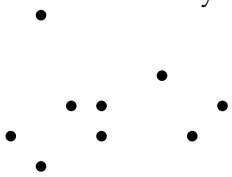
Výpis 5: Rekurzivní metoda pro vykreslení stromu

Tato metoda začíná zavoláním metody ConstructMainTree s parametrem hlavního stromu tree. Metoda pak volá pro své podstromy sama sebe s upravenými parametry, dokud má daný podstrom nějaké potomky. Konkrétně tyto 2 metody se starají o vykreslení stromu podle návrhu v kapitole 4.2.2 v části Zadání stromem → 1. Tvorba stromu. Vykreslení je na obrázku č. 7.

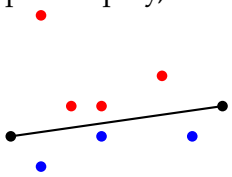
5.5.2 Convex hull algoritmus

Množina dosažitelných řešení u kooperativních her s nepřenosnou výhrou je určena konvexní obálkou. Pro určení konvexní obálky existuje mnoho metod. Pro implementaci jsem si zvolil metodu QuickHull. Tato metoda využívá algoritmus Rozděl a panuj, což je algoritmus, který řeší problém rozdělením na dílčí části (podproblémy). Tento algoritmus bude také obsahovat rekurzi. Algoritmus si můžeme zapsat do následujících kroků [7]:

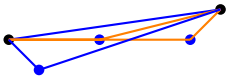
1. Nalezení bodů s nejmenší a s největší souřadnicí X. Tyto body jsou vždy součástí konvexní obálky.



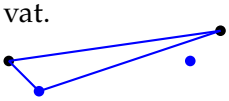
2. Pomocí pomocné čáry spojující body s předchozího kroku se body rozdělí na 2 podskupiny, které se budou zpracovávat rekurzivně.



3. Nalezení bodu s největší vzdáleností od pomocné čáry v jednom směru.

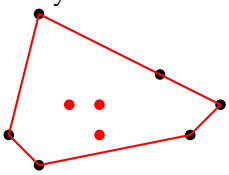


4. Body ležící uvnitř nemohou být součástí konvexní obálky, proto je budeme ignorovat.



5. Opakujte krok 3 a 4 na nově vzniklých přímkách.

6. Pokračujte tak dlouho, dokud nezbydou žádné body. Rekurze je u konce a zbylé body tvoří konvexní obálku.



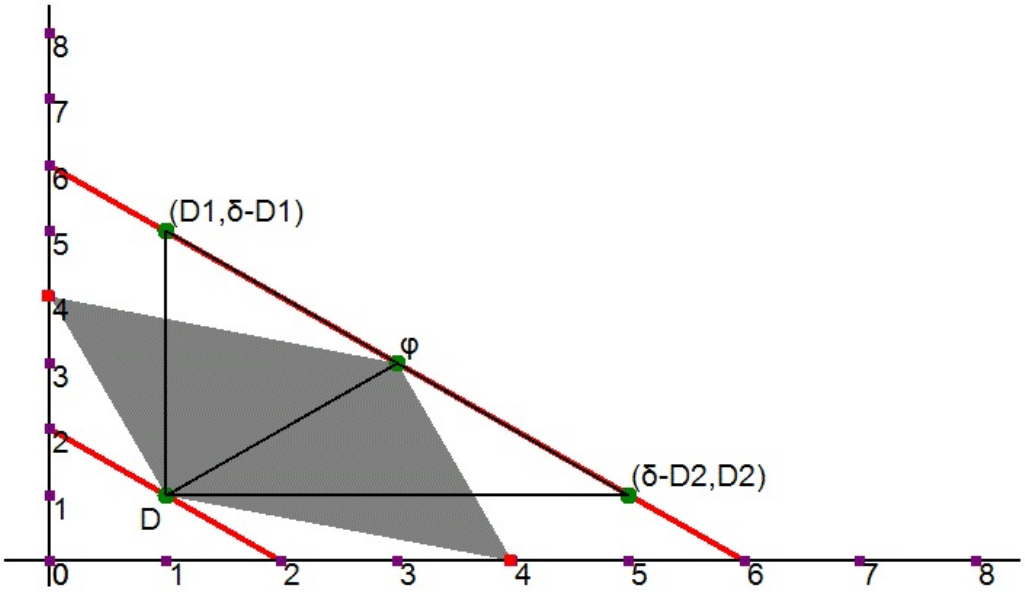
Algoritmus konvexní obálky je implementován pomocí následujících metod:

- **HullMultiToSingle** - metoda pro převod dvojrozměrné matice na jednorozměrné pole bodů.
- **getPointDirection** - metoda pro získání směru, vrací 1 nebo -1 podle toho, na které straně přímky se bod nachází.
- **getPointDistance** - metoda pro získání vzdálenosti bodu od přímky.
- **quickHull** - metoda pro první 2 kroky algoritmu. Poté volá rekurzivní metodu hullSet na obě podskupiny.
- **hullSet** - metoda provádí zbylé kroky algoritmu, dokud jí zbývají nějaké body.

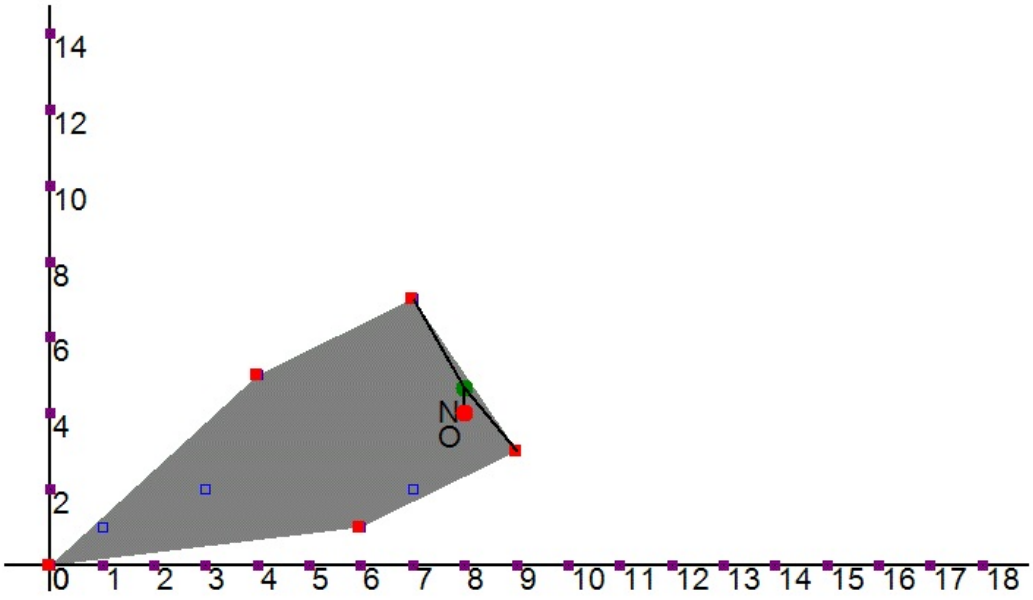
Na obrázcích 9 a 10 následuje ukázka využití konvexní obálky v aplikaci pro kooperativní hry s přenosnou a nepřenosnou výhrou.

5.5.3 Průchod stromem

Ve třídě NTree jsem často využil algoritmus průchodu stromem s Preorder průchodem, kdy se napřed projde kořen a poté se rekurzivně prochází každý podstrom tohoto kořenu ve směru zleva doprava. Následuje ukázka kódu pro hledání podstromu dle zadaného id. V této ukázce byl použit Preorder průchod stromem [8]:



Obrázek 9: Kooperativní hry s přenosnou výhrou



Obrázek 10: Kooperativní hry s nepřenosnou výhrou

```
public NTree findTree(int id)
{
    if (this.id == id)
    {
        return this;
    }
    else
    {
        foreach (var child in this.children)
        {
            NTree result = child.findTree(id);
            if (result != null)
                return result;
        }
    }
    return null;
}
```

Výpis 6: Průchod stromem Preorder

Stromem se prochází rekurzivně do té doby, dokud není nalezen podstrom s hledaným id. V případě průchodu celým stromem bez nalezení podstromu s daným id vrací funkce hodnotu null.

5.6 Hry

V rámci diplomové práce je naimplementováno pět základních úloh pro hry s nenulovým součtem. Každou z těchto her je možné analyzovat, zahrát si ji proti počítači a provést simulaci daného počtu her dvou hráčů.

5.6.1 Jednoduchý Poker

První hrou je Jednoduchý Poker, což je převzatý ukázkový řešený příklad z materiálů k Teorii her vyučované na VŠB. Jedná se o velmi jednoduchou variantu karetní hry typu Poker (Meyerson's card game). Na začátku uvažujme množinu karet, které mohou být roztříděny do dvou stejných skupin na červené a černé s pravděpodobností na sejmutí karty $\frac{1}{2}$. Vítězná karta pro prvního hráče je červená a pro druhého černá. Hra začíná u obou hráčů vložením 1 dolaru do banku. První na řadě je první hráč, který se může rozhodnout, zda zvýší o dolar a nebo položí karty. Pokud se rozhodnul zvýšit, tak je na řadě druhý hráč, který se rozhoduje zda sázku dorovná a nebo zahodí karty.

Ukázka hry Jednoduchý Poker včetně celkového vzhledu aplikace se nachází na obrázku č. 11. Na obrázku můžete vidět možnost analýzy hry pomocí 3 tlačítek. Uprostřed je aktuálně dohraná hra a ve spodní části obrazovky je možnost spustit simulaci.

5.6.2 Bitva pohlaví

Jedná se o známou ukázkovou hru z teorie her. Manželský pár má na večer domluvenou návštěvu koncertu, ale nemohou se rozhodnout jestli jít na Bacha nebo Stravinského.

Obrázek 11: Jednoduchá hra poker

Muž má raději Bacha a žena Stravinského. Pokud půjdou oba na Bacha, pak muž získá spokojenost 2 a žena 1. Pokud půjdou oba na Stravinského, pak muž získá spokojenost 1 a žena 2. Pokud oba zvolí jinou možnost, tak si koncert večer neužijí a zisk obou bude roven nule.

V rámci prodloužení hry proti počítači jsem přidal ke hře, že se stane vítězem ten, kdo získá spokojenost o hodnotě 10 a více jako první. Náhled hry se nachází na obrázku č. 12.

5.6.3 Vězňovo dilema

Vězňovo dilema je další z velmi známých her. Policie zadržela dva vězně, označme si je jako A a B. Pokud budou oba vězni mlčet, tak budou odsouzeni na kratší dobu za drobné přestupky. Pokud bude jeden mlčet a druhý vypovídat, pak ten, co vypovídal, bude osvobozen a ten, co mlčel, bude odsouzen na hodně dlouho dobu. Pokud se rozhodnou vypovídat oba dva, tak budou odsouzeni na středně dlouhou dobu. Přesné ohodnocení doby odsouzení je v tabulce č. 2.

Obrázek 12: Bitva pohlaví

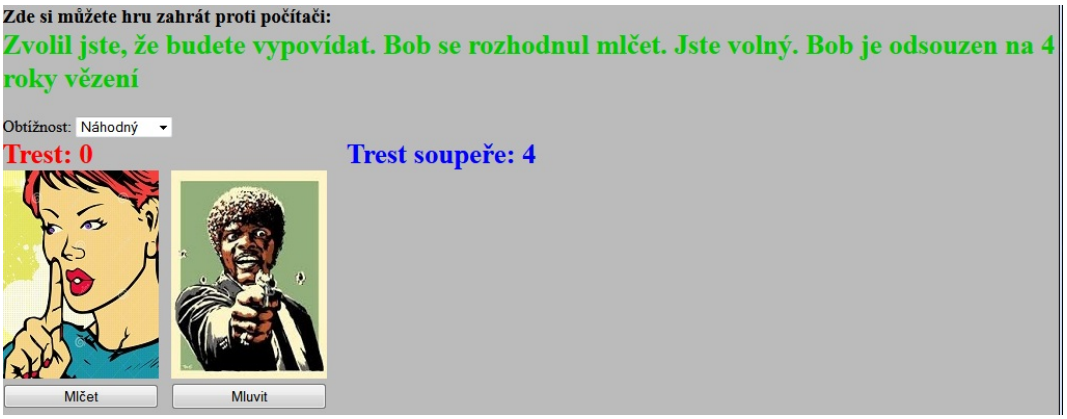
	Vězeň B mlčí	Vězeň B vypovídá
Vězeň A mlčí	Oba odsoudí na 1 rok	Vězeň A bude odsouzen na 4 roky
Vězeň A vypovídá	Vězeň B bude odsouzen na 4 roky	Oba odsouodí na 3 roky

Tabulka 2: Vězňovo dilema

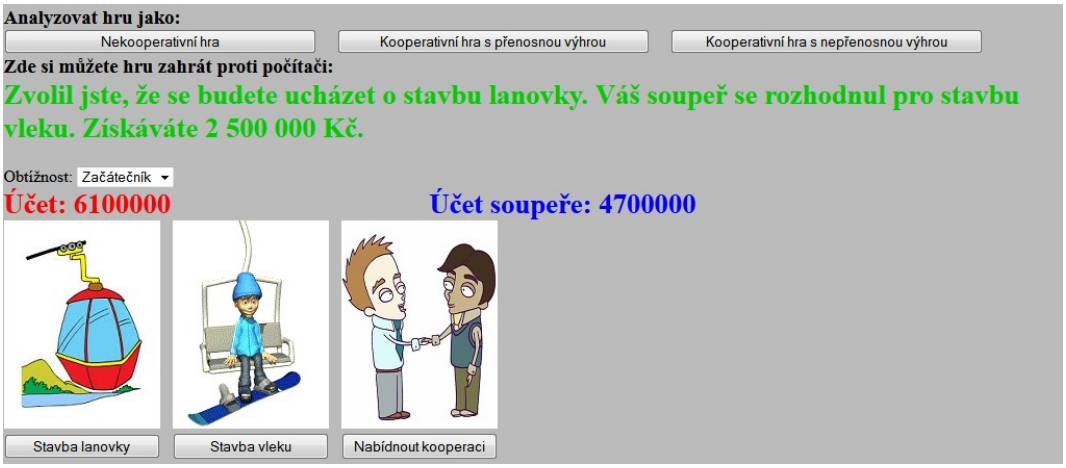
Na obrázku č. 13 se nachází ukázka aplikace.

5.6.4 Výběrové řízení

Pro následující 2 příklady jsem se nechal inspirovat hrami ze stránek ČVUT [9] z předmětu Teorie her a optimální rozhodování. Prvním z těchto příkladů je výběrové řízení. Majitel horského hotelu chce postavit lanovku a vlek. U lanovky je kalkulován zisk ve výši 2 500 tisíc korun. U zakázky na lyžařský vlek je kalkulován zisk 1 100 tisíc korun. O zakázku na výstavbu se ucházejí 2 firmy A a B. Každá firma se musí rozhodnout, zda bude stavět



Obrázek 13: Vězňovo dilema



Obrázek 14: Výběrové řízení

lanovku nebo vlek a nebo zkusí nabídnout kooperaci na obou stavbách. Vše se dále řídí následujícími pravidly:

1. Když se o jednu ze staveb bude ucházet jedna firma, získá celou tuto zakázku.
2. Když se o jednu stavbu budou ucházet obě firmy a o druhou žádná, nabídne zadavatel kooperaci oběma firmám na obou stavbách s tím, že se o provedení prací i o zisky podělí stejným dílem.
3. Když se jedna z firem uchází o stavbu jednoho zařízení a druhá o kooperaci na obou, získá firma, která nabízí realizaci celé stavby lanovky 60% a druhá firma 40% a u vleku získá firma, která nabízí realizaci celé stavby 70% a druhá 30%. Na zbývající stavbě pak firmy kooperují stejným dílem a o zisk se dělí rovněž stejným dílem.

V pravidlech hry je doplněno, že zvítězí ta firma, která dříve vydělá 20 milionů korun.

Hru můžeme převést na následující bimatici:

$$X = \begin{pmatrix} (18,18) & (25,11) & (\frac{41}{2},\frac{31}{2}) \\ (11,25) & (18,18) & (\frac{101}{5},\frac{79}{5}) \\ (\frac{31}{2},\frac{41}{2}) & (\frac{79}{5},\frac{101}{5}) & (18,18) \end{pmatrix}$$

Pro zjednodušení jsou částky v bimatici zadány ve stovkách tisíc korun. Hra je již v tomto stavu uložena v aplikaci a kliknutím na libovolnou analýzu se přejde k analýze takto zadané hry ve formě Kuhnova stromu. Na obrázku č. 14 je ukázka hry.

5.6.5 Distribuce letáků

Poslední hrou je Distribuce letáků. Dva velké obchodní řetězce postavily nákupní centra ve městě a usilují o zvýšení svého obratu distribucí reklamních letáků. Ve městě jsou očekávané zisky 100 milionů a v obcích 75 milionů. Oba řetězce mají částku X peněz

Investováno	Zisk zvýšen
X/2	+10%
X	+ 25%
3X/2	+ 50%
2X	+ 65%

Tabulka 3: Distribuce letáků - očekávané zisky

a musí se rozhodnout, zda budou letáky distribuovat ve městě nebo v obcích a nebo ve městech i obcích, ale v omezeném množství. Zisk podle způsobu distribuce se řídí následujícími pravidly:

1. Když letáky distribuuje v dané oblasti pouze 1 obchodní řetězec, pak získá celý zisk z této oblasti.
2. Když budou do distribuce oba obchodní řetězce v jedné oblasti investovat celou částku (X), pak si rozdělí zisky z této oblasti, ale přijdou tím o všechny zisky z oblasti, kde žádná kampaň neprobíhá.
3. Když budou oba obchodní řetězce investovat v každé oblasti polovinu částky (X/2), pak si rozdělí zisky z obou oblastí a navíc díky spolupráci ušetří náklady na distribuci a zbylé peníze budou moci investovat do vylepšení svých webových stránek, což každému přinese zisk navíc v hodnotě 5 milionů.
4. Když první obchodní řetězec investuje do jedné z oblastí polovinu částky (X/2) a druhý celou částku (X), pak získá první řetězec 1/4 zisků v této oblasti a druhý řetězec 3/4 zisků v této oblasti. Ve druhé oblasti získá první řetězec 1/2 zisků a druhý řetězec žádný zisk.

Dále se očekávané zisky v každé z obou oblastí mění podle investovaných peněz - viz. tabulka č. 3.

Tady už se jedná o složitější příklad. Ukážeme si jak toto zadání převést na bimatici. Postup bude zobrazen pro strategii prvního hráče r_1 a strategie druhého hráče s_1, s_2 a s_3:

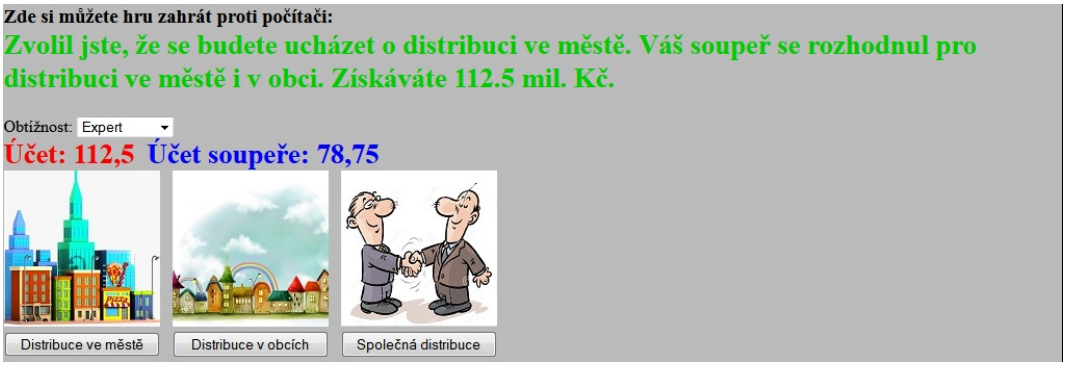
$$X_{r_1,s_1} = \left(\frac{100 + 100 \cdot 0.65}{2}, \frac{100 + 100 \cdot 0.65}{2}\right) = (82.5, 82.5)$$

$$X_{r_1,s_2} = (100 + 100 \cdot 0.25, 75 + 75 \cdot 0.25) = (125, 93.75)$$

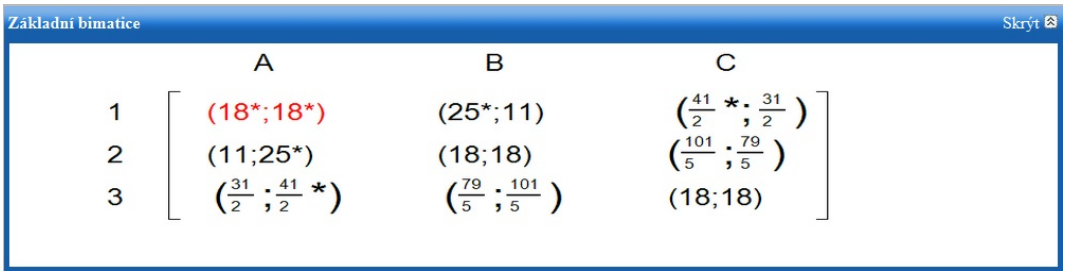
$$X_{r_1,s_3} = \left((100 + 100 \cdot 0.5) \cdot \frac{3}{4}, (100 + 100 \cdot 0.5) \cdot \frac{1}{4} + (75 + 75 \cdot 0.1) \cdot \frac{1}{2}\right) = (112.5, 78.75)$$

Z výpočtu můžeme vypozorovat, že se nejprve zjistí, kolik se do dané oblasti celkem investovalo peněz a o kolik se tím v oblasti zvýšil zisk. S touto částkou pak pracujeme dále a řídíme se hlavními 4 pravidly hry. Tímto způsobem vypočítaná matice vypadá následovně:

$$X = \begin{pmatrix} (82.5, 82.5) & (125, 93.75) & (112.5, 78.75) \\ (93.75, 125) & (61.9, 61.9) & (84.375, 83.125) \\ (78.75, 112.5) & (83.125, 84.375) & (114.375, 114.375) \end{pmatrix}$$



Obrázek 15: Distribuce letáků



Obrázek 16: Základní bimatice pro nekooperativní hry

Tato hra vychází zajímavě. Pro nekooperativní hry vychází strategie prvního hráče i druhého hráče $p = \left(\frac{3}{7}, \frac{4}{7}, 0\right)$. V případě her kooperativních s přenosnou výhrou naopak vychází, že se hráčům vyplatí spolupracovat. Ukázka hry se nachází na obrázku č. 15.

5.7 Analýza her

Pokud jde uživateli pouze o analýzu her, tak má možnost si v analýze her vytvořit vlastní hru zadanou stromem nebo maticí a nebo upravit stávající hry a pozorovat, jaký vliv mají změny na strategie a celkové řešení.

Součástí implementace je také ošetření špatných vstupních proměnných.

5.7.1 Nekooperativní hry

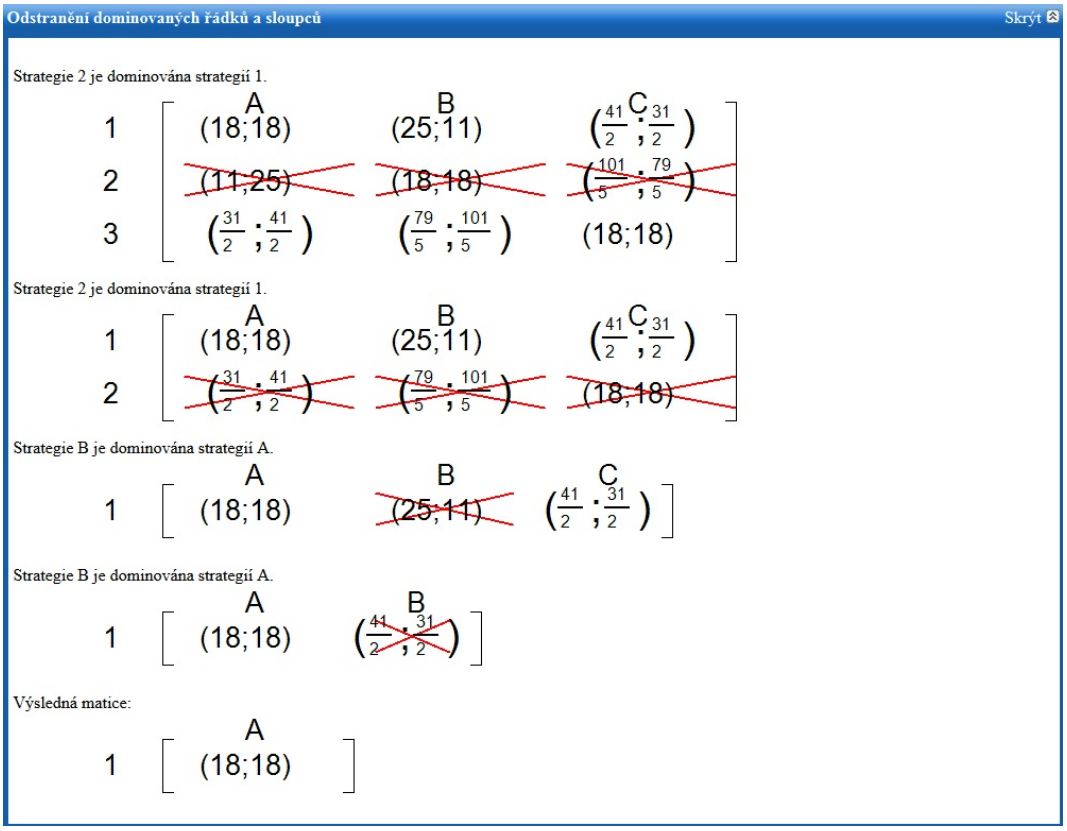
Nekooperativní hry se skládají z následujících sekcí:

- **Zadání stromem a zadání tabulkou** - implementováno dle kapitoly 4.2.2
- **Základní bimatice** - vykreslení bimatice ze stromu nebo tabulky včetně zvýraznění Nashových rovnovážných bodů. Ukázka vykreslené bimatice je na obrázku č. 16.
- **Nashovy rovnovážné body** - v této sekci jsou vypsány výše nalezené Nashovy rovnovážné body.

- **Matice hráčů** - v této sekci je bimatice převedena na 2 matice.
- **Odstranění dominovaných řádků a sloupců** - dominance je jedna ze zajímavých funkcí. Metoda pro výpočet dominance bimatice se nazývá DominationNonZero-Sum. Vzhledem k tomu, že je velmi obsáhlá, bude popsána pseudokódem:

```
SEZNAM DominateNonZeroSum(MATICE, PRESNOST)
{
SKOK:
    projdi  RADKY od zacatku do konce //porovnani radku
    {
        projdi  RADKY od radku z predchoziho cyklu do konce
        {
            JEDOMINOVANO nastav na FALSE
            projdi  SLOUPCE
                pokud naleznes dominovany RADEK, pak nastav JEDOMINOVANO
                    na true
            pokud JEDOMINOVANO
            {
                vykresli  aktualni  MATICI
                smaz RADEK
                skoc na SKOK
            }
        }
    }
    //  PROVED TO SAME PRO SLOUPCE
    projdi  RADKY od zacatku do konce //porovnani kombinaci radku
    {
        projdi  RADKY od radku z predchoziho cyklu do konce
            projdi  RADKY od radku z predchoziho cyklu do konce
                projdi  cyklus od 0 po PRESNOST
                {
                    JEDOMINOVANO nastav na FALSE
                    projdi  SLOUPCE
                        udelej  kombinaci vsech trojic  RADKU (3 porovnani) v
                            pomeru podle PRESNOST a pokud naleznes
                            dominovany RADEK, pak nastav JEDOMINOVANO na
                            true
                    pokud JEDOMINOVANO
                    {
                        vykresli  aktualni  MATICI
                        smaz RADEK
                        skoc na SKOK
                    }
                }
    }
    //  PROVED TO SAME PRO SLOUPCE
    vykresli  aktualni  MATICI
    vrat  SEZNAM obrazku
}
```

Výpis 7: Dominance - pseudokód



Obrázek 17: Odstranění dominovaných řádků a sloupců

Algoritmus nejdříve porovná vzájemně všechny řádky a poté všechny sloupce. Při nalezení dominovaného řádku nebo sloupce jej odstraní a příkazem skoku pokračuje v algoritmu od začátku až do té doby, dokud nebude v bimatici žádný dominovaný řádek ani sloupec. Algoritmus je dále rozšířen o smíšenou dominanci, kdy se řádky či sloupce místo ve 2 na sobě navazujících for cyklech projdou ve 3 na sebe navazujících for cyklech. Poté dojde k vzájemnému porovnání jednoho řádku či sloupce s kombinací dvou jiných v poměru, který je určen pomocí proměnné PRESNOST. V programu je nastavena proměnná PRESNOST na hodnotu 100 což znamená, že zbylé dva sloupce či řádky se kombinují postupně v poměru od 1:99, 2:98 až po 99:1. Zvýšením přesnosti stoupá čas potřebný k výpočtu.
Ukázka dominance pro hru Výběrové řízení je na obrázku č. 17.

- **Optimální strategie a bezpečná úroveň** - v této sekci se nachází výpočet optimální strategie, bezpečné úrovně hry a ekvalizující strategie.

5.7.2 Kooperativní hry s přenosnou výhrou

Dělí se na následující sekce:

- **Zadání stromem a zadání tabulkou**
- **Základní bimatice** - vykreslení bimatice ze zadaného stromu nebo tabulky
- **Grafický náhled** - grafické řešení hry. Nachází se na obrázku č. 9. Algoritmicky řešeno pomocí konvexní obálky a přímek $y = -1$ protínajících nejvzdálenější a nejbližší bod konvexní obálky
- **Matice hráčů**
- **Strategie hrozby a cena hry** - v této sekci se nachází výpočet strategií hrozeb obou hráčů a ceny hry
- **Řešení** - zde se nachází kooperativní strategie, rozdělení výher, platba bokem a bod nedohody. Za zmínku stojí bod nedohody, který se spočítá podle následujícího vztahu:

  $$D = D(p, q) = (p^T \cdot A \cdot q, p^T \cdot B \cdot q) = (D_1, D_2)$$

  Algoritmicky je to řešené pomocí klasického algoritmu pro násobení vektoru s maticí a následném násobení nově vzniklého vektoru s vektorem.

5.7.3 Kooperativní hry s nepřenosnou výhrou

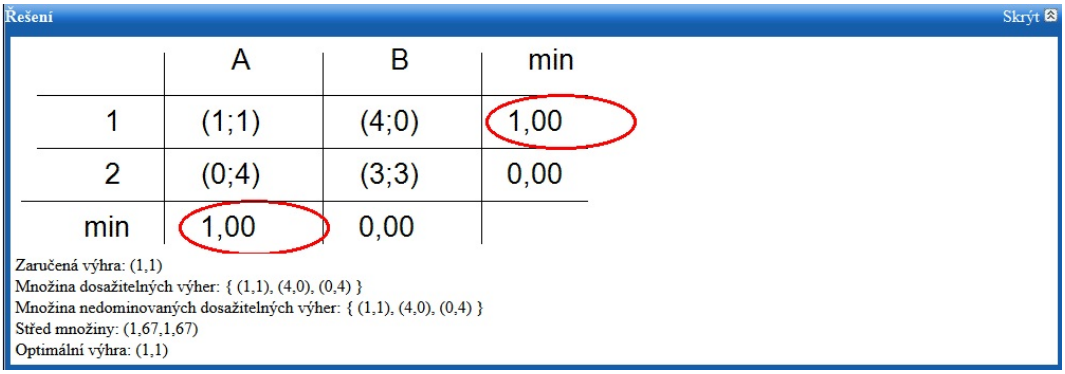
Rozdělení je následující:

- **Zadání stromem a zadání tabulkou**
- **Základní bimatice** - vykreslení bimatice jako u her s přenosnou výhrou
- **Grafický náhled** - nachází se na obrázku č. 10. Množinou dosažitelných řešení je v tomto případě přímo konvexní obálka.
- **Matice hráčů**
- **Řešení** - nachází se zde výpočet zaručené výhry, množiny dosažitelných výher, množiny nedominovaných dosažitelných výher, střed množiny a optimální výhra. Ukázka pro řešení hry Vězňovo dilema se nachází na obrázku č. 18.

5.8 Simulace hry

Jak již bylo zmíněno v návrhu aplikace, tak je nejprve nutné zvolit obtížnost prvního a druhého hráče. Poté se zvolí počet her k odehrání a spustí se simulace. Počet simulací je omezen na hodnotu 1 000 000. Je to dostatečný počet k získání věrohodných výsledků a zároveň se dočkáme velmi rychle výsledků simulace.

Máme celkem 5 obtížností hráčů: Náhodný, Začátečník, Pokročilý, Expert, Optimální. Obtížnost hráče je určena tím, v jakém poměru volí mezi náhodnou a optimální strategií.

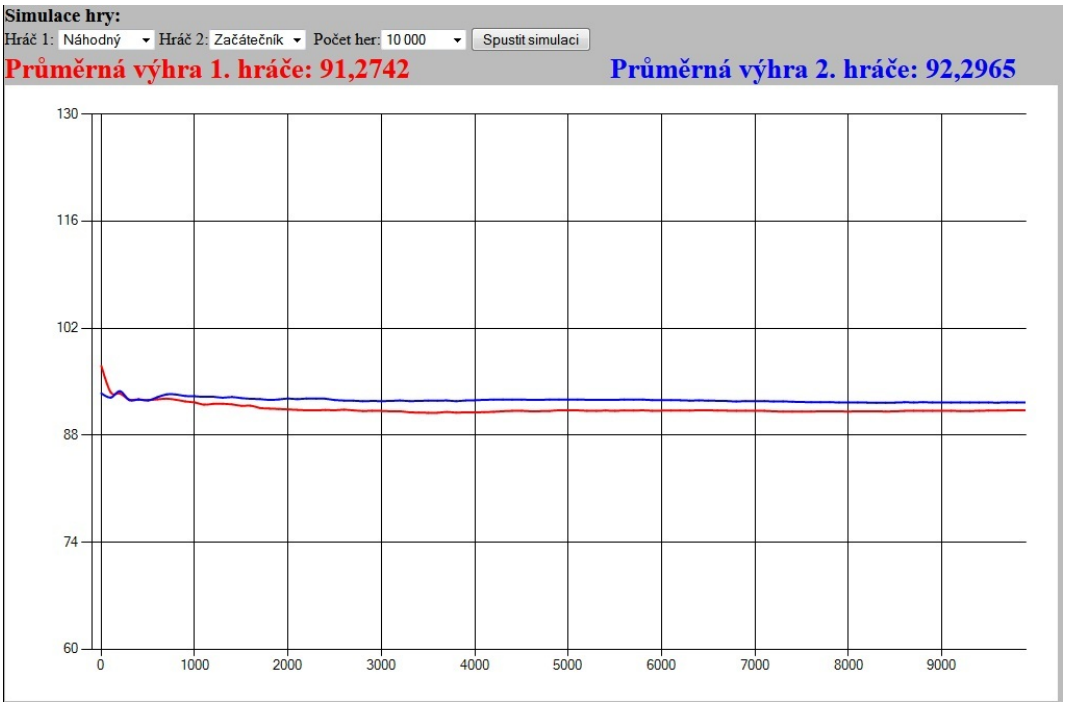


Obrázek 18: Řešení kooperativní hry s přenosnou výhrou Vězňovo dilema

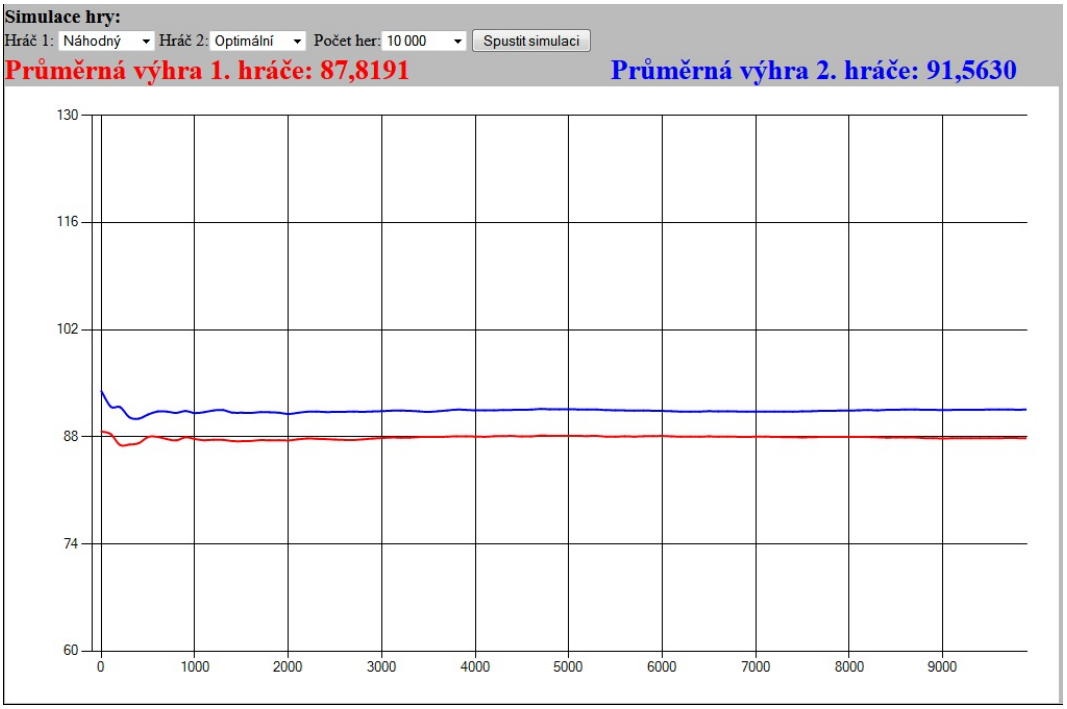
Už podle názvu je patrné, že hráč hrající podle obtížnosti Náhodný hraje naprosto náhodně a hráč hrající podle obtížnosti Optimální hraje vždy optimální strategii. Zbývající obtížnosti jsou nastaveny na 25%, 50% a 75% z optimální strategie.

Na obrázku č. 19 následuje ukázka simulace náhodného hráče proti začátečníkovi ve hře Distribuce letáků pro 10 000 her.

S roustoucí obtížností roste i zisk daného hráče. O čemž se můžeme přesvědčit na obrázku č. 20, na kterém je zobrazena stejná hra jen proti optimální obtížnosti.



Obrázek 19: Simulace hry - Náhodný vs Začátečník



Obrázek 20: Simulace hry - Náhodný vs Optimální

6 Testování a nasazení serveru

Předposlední kapitola se týká testování serveru a jeho nasazení do reálného provozu.

6.1 Testování

Důležitou součástí vývoje aplikace je testování. Aplikaci jsem testoval sám během implementace. K testování funkčnosti aplikace také docházelo na pravidelných konzultacích s vedoucím diplomové práce. Testování aplikace jsem rozdělil na tyto části:

- **Testování GUI** - grafické rozhraní aplikace bylo otestováno pro různé vstupy programu. Důraz jsem kladl především na správné vykreslení a zarovnání všech objektů v aplikaci.
- **Testování pro různé prohlížeče** - v rámci splnění nefunkčních požadavků v kapitole 3.2.4 je aplikace otestována v těchto nejznámějších prohlížečích:
  - Microsoft Internet Explorer 8
  - Mozilla Firefox 28.0
  - Opera 18
  - Google Chrome 34
- **Testování hraní her a simulace** - všechny naimplementované hry byly řádně otestovány proti různým nastavením obtížností.
- **Testování analýzy her** - k testování analýzy her jsem využil příklady z výukových textů k předmětu Game Theory vyučovaném na University of California, Los Angeles. Dále příklady z veřejně přístupných materiálů ze stránek předmětu Teorie her vyučovaném na VŠB v Ostravě. [10]

6.2 Nasazení

Pro testování je možné aplikaci pouštět lokálně v prostředí Microsoft Visual Studio. K nasazení do reálného provozu je třeba mít hosting s podporou technologie ASP.NET. Pro ukázku běžící aplikace jsme zvolili adresu *http://teh-vsb.aspone.cz/* od poskytovatele webhostingu zdarma ASPone.

K nasazení na server stačí celý projekt publikovat v Microsoft Visual Studiu a následně nově vzniklé soubory nakopírovat přes FTP klienta na webhosting. V případě problémů s instalací doporučuji projít si video návody, diskuzi nebo případně kontaktovat technickou podporu daného webhostingu.

7 Závěr

Hlavním cílem této série diplomových prací bylo vytvoření serveru pro podporu výuky předmětu Teorie her a pomoci tak studentům studujícím tento předmět k lepšímu pochopení dané problematiky. Mým cílem bylo tuto sérii rozšířit o oblast her s nenulovým součtem.

Důležitou součástí práce bylo pochopení dané oblasti teorie her. Základní potřebné znalosti jsem získal absolvováním předmětu Teorie her na VŠB v Ostravě u pana doc. Ing. Zdeňka Sawy, Ph.D. Další znalosti jsem nabyl samostudiem skript a pomocí internetu. Při práci na projektu jsem implementoval především ty části teorie her, které se probíraly během výuky.

Na začátku projektu jsem začal spolupracovat s kolegou Martinem Hurábem, dohodli jsme se na architektuře a vzhledu aplikace, neboť některé části projektu nás obou byly velmi podobné a bylo proto vhodné doržet jednotný vzhled a funkčnost aplikace. Já jsem pracoval na vývoji Kuhnova stromu, kolega na řešení her s nulovým součtem a navzájem jsme si daná řešení poskytli.

Aplikace nyní umožňuje nastudovat si potřebnou látku k pochopení učiva. Dále je možné provést analýzu podle zadání nekooperativních her, kooperativních her s přenosnou výhrou a kooperativnách her s nepřenosnou výhrou. Návštěvníci stránek mají také možnost zahrát si pět ukázkových her proti počítači podle zvolené obtížnosti a vyzkoušet si simulovanou hru 2 hráčů také podle zadaných obtížností a se zadaným počtem odsimulovaných her.

Přínosem této práce je zlepšení výuky Teorie her pro studenty. Pro mne osobně bylo přínosem, že jsem si díky této diplomové práci prohloubil znalosti vývoje webových aplikací pod technologií ASP.NET. Dále jsem si rozšířil své programátorské znalosti v rámci implementace různých algoritmických problémů jako jsou: konvexní obálka, rekurze a průchod stromem a také jsem získal zkušenosti ohledně vývoje aplikace založené na společné architektuře.

Celý server je nyní připraven k nasazení do skutečného provozu. Do budoucna je možné rozšířit stávající počet úloh, případně naimplementovat detailnější analýzu řešení her.

8 Reference

[1] HURÁB, Martin, Server pro podporu výuky teorie her. Ostrava, 2014 [cit. 2014-05-01]. Diplomová práce. VŠB - Technická univerzita Ostrava.

[2] KUBÍČEK, Michal. Velký průvodce SEO: jak dosáhnout nejlepších pozic ve vyhledávačích. Vyd. 1. Brno: Computer Press, 2008, 318 s. ISBN 978-80-251-2195-5.

[3] StatCounter, GlobalStats [online]. 1999-2014 [cit. 2014-05-01]. *Dostupné z: http://gs.statcounter.com/*

[4] Product Documentation - Embarcadero Technologies [online], [cit. 2014-05-01]. *Dostupné z: http://zaskodny.cz/embarcadero.php*

[5] BAŘÁK, Tomáš, Server pro podporu výuky teorie her. Ostrava, 2013 [cit. 2014-05-04]. Diplomová práce. VŠB - Technická univerzita Ostrava.

[6] NAGEL, Christian. Professional C# 4 and .Net 4. Vyd. 1. Indianapolis, IN: Wiley Pub., c2010, 1474 p. ISBN 978-0470502259.

[7] Wikipedia, The Free Encyclopedia, Convex Hull algorithms [online]. [cit. 2014-04-18]. *Dostupné z: http://en.wikipedia.org/wiki/Convex_hull_algorithms*

[8] Algoritmy.net [online], [cit. 2014-04-26]. *Dostupné z: http://www.algoritmy.net/article/104/Strom*

[9] HYKŠOVÁ, Magdalena. Teorie her: Teorie her a optimální rozhodování. [online]. [cit. 2014-04-27]. *Dostupné z: http://euler.fd.cvut.cz/predmety/teorie_her/*

[10] FERGUSON, Thomas S., Game Theory [online]. 2005 [cit. 2014-04-20]. *Dostupné z: http://www.math.ucla.edu/ tom/math167.html*

9 Přílohy

1. **Uživatelský manuál** - je dostupný na adrese *http://teh-vsb.aspone.cz/NonZeroSumUserGuide.aspx*
2. **CD** - obsahuje diplomovou práci, dokumentaci a zdrojové kódy celé aplikace
   - **documentation** - dokumentace k aplikaci
   - **projekt** - řešení projektu, které se dále dělí na:
     - **NonZeroSum** - projekt pro logiku her s nenulovým součtem
     - **UI** - projekt pro GUI aplikace
     - **ZeroSum** - projekt pro logiku her s nulovým součtem
   - **publish** - publikovaný projekt, který je připraven k nasazení na server
   - **tex** - zdrojové soubory pro LaTeX včetně obrázků
   - **text** - text diplomové práce